Universidad Nacional de Educación a Distancia

Escuela Técnica Superior de Ingeniería Informática

Máster Universitario en Ciberseguridad

Curso 2021/2022

Análisis del ransomware Ryuk, impacto en los sistemas afectados y técnicas API Hooking y de aprendizaje automático como medidas de mitigación ante el ransomware.

Héctor Santamaría Claro

Trabajo de fin de máster

UNED

Análisis del ransomware Ryuk, impacto en los sistemas afectados y técnicas API Hooking y de aprendizaje automático como medidas de mitigación ante el ransomware.

Autor: Héctor Santamaría Claro
Tutor: Antonio Robles Gómez

Curso 2021-2022
Convocatoria septiembre

Resumen

La amenaza del ransomware se ha convertido en los últimos años en motivo de preocupación para empresas y usuarios. Algunos ataques han implicado costes millonarios, así como pérdida de información valiosa. Este trabajo se centra en el ransomware y en las particularidades que giran en torno a este tipo de software malicioso, poniendo el foco en Ryuk, ransomware muy sofisticado que comparte características comunes con otros programas utilizados con los mismos fines.

Primeramente, se exponen de forma breve ciertos métodos que los ciberdelincuentes llevan a cabo para ocultar y distribuir malware mostrando ejemplos de su implementación. Después de esto, comienza un análisis riguroso de una muestra de Ryuk: estático, dinámico y de código; se recopila información que revela la forma de interactuar con el sistema operativo, el movimiento lateral o propagación y el cifrado de archivos.

Aprovechando las averiguaciones con respecto a las capacidades criptográficas de la muestra analizada y otros tipos de ransomware, se crea un programa capaz de monitorizar los procesos del sistema mediante API Hooking. Este desarrollo puede detectar procesos que realicen llamadas a determinadas funciones de Windows para bloquearlos, extraer información relevante para un analista o recuperar, en algunas implementaciones, las contraseñas de cifrado generadas.

Finalmente, se realiza un estudio sobre la generación de modelos predictivos para detectar ransomware mediante dos algoritmos de aprendizaje automático, proveyéndose de características estáticas comunes a cualquier fichero Portable Ejecutable.

Palabras clave

Ransomware, Ryuk, CryptoAPI, API Hooking, aprendizaje automático, Árboles de Decisión.

Summary

The threat of ransomware has become a cause for concern for companies and users in recent years. Some attacks have involved costs in the millions, as well as loss of valuable information. This paper focuses on ransomware and the particularities surrounding this type of malware, focusing on Ryuk, a very sophisticated ransomware that shares common characteristics with other programs used for the same purposes.

First of all, certain methods that cybercriminals use to hide and distribute malware are briefly presented, showing examples of their implementation. This is followed by a rigorous analysis of a sample of Ryuk: static, dynamic and code; information is gathered that reveals how it interacts with the operating system, lateral movement or propagation and file encryption.

Taking advantage of the findings regarding the cryptographic capabilities of the analyzed sample and other types of ransomware, a program capable of monitoring system processes using API Hooking is created. This development can detect processes that make calls to certain Windows functions in order to block them, extract relevant information for an analyst or recover, in some implementations, the encryption passwords generated.

Finally, a study is made on the generation of predictive models to detect ransomware using two machine learning algorithms, providing them with static features common to any Portable Executable file.

Keywords

Ransomware, Ryuk, CryptoAPI, API Hooking, Machine Learning, Decision Tree Algorithms.

Índice

Índice de Figuras

Índice de tablas

1. Introducción

El ransomware “ransom” (rescate en español) y “ware” (producto), es un tipo de software malicioso que impide que los usuarios afectados accedan a los recursos de sus dispositivos informáticos y a datos personales, normalmente cifrándolos. (Nihad A. Hassan, 2019). La intencionalidad de los autores de ransomware es conseguir un beneficio económico a cambio de desencriptar los archivos y datos cifrados de los usuarios u organizaciones cuyos sistemas se han visto afectados.

Año tras año el número de ataques y pérdidas económicas va en aumento y en el mercado negro se distribuye el ransomware como servicio (RaaS), donde otros ciberdelincuentes contratan un programa de ransomware para realizar campañas durante un tiempo determinado y dividen las ganancias obtenidas con los creadores o dueños. Estos kits de ransomware se pueden contratar por unos cientos o miles de dólares en la “Dark Web” y ofrecen posibilidades de pago en criptomonedas para dificultar el rastreo y posterior captura de los responsables por parte de las autoridades.

Algunas de las vías que se suelen utilizar para llegar a ejecutar malware en un dispositivo son similares a ataques correspondientes con otros tipos de software malicioso:

- Mediante técnicas de ingeniería social o diccionarios de contraseñas, conseguir claves de acceso (nombres de usuarios y contraseñas) para poder entrar en sistemas con permisos administrativos, hacer uso de protocolos como RDP (Protocolo de Escritorio Remoto de Microsoft) y SMB (Server Message Block) o bien facilitan la propagación entre otros computadores, o sirven para dejar expuestos en la red potenciales objetivos con las que llevar a cabo ataque de fuerza bruta u otras técnicas más sofisticadas.

- Engañar a los usuarios para que ejecuten un archivo o abran un enlace que contiene el ransomware desde correos electrónicos falsos o que suplantan la identidad dentro del marco del phishing y el spam.

- Valerse del navegador para que la víctima visite sitios web, descargue y ejecute un archivo malicioso creyendo que se trata de un fichero inofensivo. Estos sitios web también

podrían ser legítimos y estar infectados con contenido malicioso que se encargaría de analizar el dispositivo del usuario para después insertarlo aprovechando una vulnerabilidad subyacente. Existen otros modos que explotan vulnerabilidades en los navegadores que permitirían activar el ransomware sin tener que descargar nada necesariamente, por ejemplo, mediante malvertising.

- A través de las descargas piratas y promesas de parches, cracks, keygens se sabe que es utilizado como medio de difusión para malware de todo tipo, como es el caso de algunas de las últimas campañas del ransomware DJVU. (Webb. N, 2022).

Continuamente aparecen nuevas variedades de ransomware que se hacen más o menos populares según su eficacia o el éxito del impacto de sus campañas. No obstante, en los últimos años se han ideado estrategias para mitigar las infecciones por ransomware de cara a organizaciones y usuarios, así como numerosas soluciones antivirus, IPS e HIPS, como resistencia ante estas amenazas

El tema elegido resulta interesante por la utilidad que aporta a la hora de comprender la facilidad con la que se puede distribuir ransomware; el funcionamiento y los procesos internos de este; el cifrado de archivos y documentos; cómo se propaga a otros equipos de la red y los métodos de persistencia.

No se debe olvidar que en los últimos años se han hecho avances importantes en cuanto al desarrollo de algoritmos y sistemas de aprendizaje automático aplicados a la ciberseguridad. Motivo por el cual otro de los temas de este trabajo de fin de máster, que aborda cómo hacer uso de las características comunes en un dataset de características de archivos binarios de ransomware conocidos para desarrollar un modelo predictivo, también es útil para explicar e indagar sobre estas estrategias y los recientes algoritmos de machine learning.

Lo expuesto anteriormente resume la creencia del alumno de resultar los temas que abarca el TFM útiles y adecuados para los tiempos actuales y los desafíos a los que se enfrenta la sociedad en cuanto a la ciberseguridad y los ataques con malware y, especialmente, ransomware.

1.1- Motivación y objetivos

1.1.1- Motivación

No son necesarios conocimientos especializados en ciberseguridad para entender el impacto que puede tener en una organización o ,simplemente, en un usuario doméstico la infección causada por cualquier software malicioso. El daño ejercido puede, por ejemplo, comprender modificaciones en el sistema operativo que dificulten su uso con normalidad, exfiltración de datos, que quizá deriven en consecuencias peores o pérdida de información.

En el caso del ransomware los efectos son particularmente graves. Los datos de cualquier sistema informático afectado quedan inservibles y sin una garantía clara de poder ser recuperados en caso de aceptar el chantaje. El número de ataques por estos medios se está incrementando, surgiendo nuevas variantes que implican pérdidas económicas devastadoras e, incluso, a nivel sentimental y profesional; pues muchas personas se han visto agraviadas por los ficheros perdidos.

Son estos motivos los que condicionan el interés en realizar esta investigación que habla realmente sobre malware, pero se centra en el ransomware y sus particularidades, con el objetivo de aportar datos e información que resulte útil para generar conocimiento sobre la muestra elegida, muy popular en los últimos dos años y sobre métodos alternativos para su detección y mitigación, que van más allá de los tradicionales sistemas IPS o IDS basados en firmas.

1.1.2- Objetivo general

El objetivo general de este trabajo es experimentar e investigar sobre el proceso llevado a cabo por los ciberdelincuentes desde que se distribuye ransomware de su código en el sistema operativo, la forma en la que puede encriptar los datos y el trabajo que corresponde a un experto en ciberseguridad para mitigar sus efectos o prevenir la infección.

Como parte introductoria, se expone la relativa facilidad con la que se puede propagar ransomware mediante campañas de Spear-Phishing y utilizar algunas técnicas bien conocidas

de ocultación y de ingeniería social para lograr que el usuario ejecute el ransomware en su computadora, con la creencia de que se trata de un archivo legítimo e inofensivo.

Se investigará sobre el funcionamiento y procesos empleados por el ransomware Ryuk para iniciar su actividad maliciosa en el ordenador de la víctima, el encriptado de archivos, la persistencia en el sistema operativo después de ser reiniciado y la forma en la que puede replicarse entre otros sistemas de la red. Respecto al cifrado de datos, se abordará con el desarrollo de un método que pueda servir como medida de mitigación ante los procesos criptográficos.

Finalmente, mediante un set de datos con características estáticas de ransomware y programas ejecutables benignos, diseñados ambos tipos para sistemas operativos Windows, se experimentará con dos actuales algoritmos de clasificación para crear un modelo que permita predecir con relativa exactitud si un archivo binario es ransomware o no. También se hará un análisis con las características consideradas más relevantes por el alumno y cuáles realmente lo eran más o no, según el algoritmo empleado.

1.1.3- Objetivos específicos

Para poder alcanzar el objetivo general será necesario que se verifiquen distintos objetivos específicos que se abordarán en este TFM y que son:

- Poner en práctica técnicas para ocultar ransomware o simular su apariencia inofensiva mediante el cambio de la apariencia del fichero ejecutable junto con su nombre y extensión, así como proceder a su empaquetamiento y encriptación.

- Desarrollar el proceso necesario para crear una campaña de spear-phishing que distribuya el ransomware que, junto con algunas técnicas de ingeniería social y modificación de las cabeceras del correo electrónico (email spoofing), simulen ser un mensaje legítimo y relevante para la víctima.

- Realizar un análisis estático del ransomware seleccionado para este trabajo, examinando cadenas de texto, llamadas y uso de las API (application program interface) del sistema,

contenido cifrado y otras características que puedan dilucidar las funcionalidades y propósito del malware en esta primera etapa del estudio.

- Realizar un análisis dinámico del ransomware seleccionado estudiando su comportamiento e interacción con el sistema operativo desde el momento de su ejecución.

- Llevar a cabo un análisis de código con un desensamblado o depurador, para ahondar en las particularidades y funcionamiento interno de Ryuk.

- Estudiar los métodos empleados para el cifrado de la información y cómo son alterados los archivos que modifica. Valorar si es posible revertir el proceso de cifrado o establecer alguna estrategia de mitigación con el desarrollo de una herramienta.

- Indagar sobre las propiedades estáticas del ransomware y los componentes de un fichero PE (Portable Ejecutable) para determinar cuáles son más o menos relevantes a la hora de aplicar algoritmos de aprendizaje automático.

- Desarrollar el código y los scripts necesarios para trabajar sobre un dataset con características estáticas de ransomware e implementar algoritmos de clasificación que puedan generar un modelo predictivo y llevar a cabo la predicción sobre un fichero binario cualquiera.

- Entrenar y generar los modelos extrayendo, además, información sobre la eficacia de los algoritmos utilizados y la influencia de las características seleccionadas. Posteriormente, dichos modelos serán probados con binarios reales.

1.2- Enfoque y método seguido

Los temas que se abordan tienen relación con el ransomware y el proceso que abarca desde su distribución y métodos de evasión, seguido de su impacto en los dispositivos afectados, para terminar exponiendo posibles métodos de mitigación en el cifrado de los archivos y detección temprana con modelos predictivos. Son temas variados, pero no estrictamente interrelacionados, así que la estrategia para alcanzar los objetivos varía.

Aunque es más típica del desarrollo de software, la metodología seguida se podría asemejar al modelo de ciclo de vida en cascada, donde cada etapa se lleva a cabo escalonadamente y solo cuando se termina la primera, comienza la segunda. En general, todos los apartados de este trabajo de fin de máster requieren comprender la estructura de un archivo PE (Portable Ejecutable) para sistemas Windows x86, pero en cada una de las fases hay un proceso de documentación y estudio teórico sobre las materias y áreas de conocimiento relacionadas. Inicialmente, antes de comenzar ninguna de ellas, se escogió el ransomware que el alumno consideró más favorable para las investigaciones: Ryuk.

Por estos motivos, en la memoria se pueden detallar cuatro capítulos. Cada uno aporta información útil para el siguiente.

Ocultación y envío masivo de ransomware

En el capítulo 3 se repasarán contenidos de la asignatura “Hacking Ético”, impartida en el Máster Universitario en Ciberseguridad, sobre técnicas de ingeniería social, junto con algunos ejemplos de phishing reales. También se recopilará información sobre técnicas comúnmente utilizadas para camuflar malware que se mencionan en la asignatura “Análisis de Malware”.

Cobran relevancia herramientas de código abierto para trabajar con archivos ejecutables. También software libra en la simulación del spear-phishing, servicios de red, creación de servidores de correo y análisis forense, principalmente diseñadas para sistemas Unix/Linux.

En el proceso llevado a cabo se creará una factura falsa que reclama un pago en nombre de una importante compañía. Posteriormente, se manipularán archivos PE para modificarlos

desde un punto de vista visual, empaquetarlos o encriptarlos de forman que oculten código malicioso en su interior bajo la apariencia de la falsa factura.

Esta primera parte terminará con la construcción de un servidor de correo local que permita hacer pruebas de distribución masiva de correo electrónico alterando las cabeceras del email y adjuntar el software malicioso anteriormente camuflado.

El propósito será analizar el efecto de las modificaciones en los archivos ejecutables con malware y la viabilidad de distribuirlos a través de ataques spear-phishing de la forma más creíble posible.

Análisis estático, dinámico y de código

En el capítulo 4 y capítulo 5, como fuentes de información se optará por bibliografía especializada en el análisis de malware, aprovechando los conocimientos adquiridos en la asignatura cursada del mismo nombre. En relación con la metodología, es la habitual en el análisis de malware y descrita en el material consultado.

Tal y como se ha argumentado anteriormente en los objetivos generales, el propósito de este TFM (Trabajo de fin de máster) es englobar un ciclo que transcurre desde la distribución del ransomware a las medidas de mitigación y prevención. Es importante conocer cómo puede ser la estructura de un binario ransomware (las cabeceras, sus llamadas a funciones de API, características del propio fichero, etc.) a modo de fase previa para abordar la predicción de archivos ejecutables a través de sus características estáticas. Especialmente con el análisis estático y de código de Ryuk pueden entenderse mejor la relevancia que podrían ocupar los datos de las características empleadas en el dataset.

El trabajo con los archivos binarios requerirá el uso de herramientas de software libre disponibles en la red, ampliamente conocidas y a las que se recurre habitualmente por parte de investigadores y analistas con las funcionalidades de: extracción de características, visualización del código, ejecución controlada o monitorización de la actividad en el sistema operativo.

En este segundo apartado se distinguen tres fases consecutivas en el análisis de, en este caso, el ransomware Ryuk:

- Análisis estático para obtener la máxima cantidad de información posible del binario sin llegar a ejecutarlo. Se prestará especial atención a las librerías y llamadas a funciones de Windows, así como las cadenas de texto rescatadas del código fuente.

- Análisis dinámico para obtener datos sobre su comportamiento en uno o varios sistemas operativos. Se estudiarán los efectos que tiene sobre un solo equipo y sobre una red simulada cuando entran en funcionamiento sus capacidades de movimiento lateral.

- Análisis de código para comprender de forma más precisa cómo utiliza los recursos del sistema operativo y memoria, ejecutándolo paso a paso. Aquí se aprovechará lo aprendido en las fases de análisis estático y dinámico.

Métodos criptográficos en Ryuk y otros ransomware

En el capítulo 6 se tendrán en cuenta los conocimientos adquiridos en la asignatura Criptografía Aplicada y abundante documentación en el sitio web de Microsoft sobre el funcionamiento de las APIs de Windows implicadas. Habiendo comprendido esos conceptos se planteará una posible estrategia de mitigación para registrar las claves criptográficas generadas o, al menos, bloquear el proceso de cifrado.

El procedimiento consistirá en analizar los métodos que Ryuk manejaba para cifrar los archivos de la computadora afectada y cómo afectaba a la estructura de estos. Se desarrollará una herramienta o método que permita crear un enganche hacia las llamadas de un proceso determinado a las API de Windows, de tal modo que pueda extraerse la información que se transmite en los parámetros de las funciones.

Machine Learning con características estáticas de Ransomware

En el capítulo 7 se aplicará lo aprendido en la asignatura Introducción al Aprendizaje Automático para Ciberseguridad, empleando herramientas similares, bibliotecas de software libre para los algoritmos de aprendizaje automático.

En este último apartado se seguirá una metodología muy pautada:

- Obtención de un set de datos válido con características de múltiples especímenes ransomware y archivos ejecutables benignos.

- Estudio de las características que podrían ser más relevantes a la hora de generar predicciones útiles para clasificar entre ejecutables que podrían ser ransomware o no representar ninguna amenaza.

- Entrenamientos de los modelos predictivos con dos algoritmos de clasificación, experimentando con distintas configuraciones de hiperparámetros. Se examinarán las diferencias entre ambos, tanto en la fiabilidad representada en las puntuaciones de precisión y matriz de confusión, como en las características a las que cada uno concede mayor o menor importancia.

- Una vez que los modelos presenten unos valores adecuados en los teses de entrenamiento, se harán pruebas con binarios reales.

1.3- Planificación del trabajo y presupuesto

En este apartado se propone una planificación aproximada para un proyecto como el presentado en este trabajo de fin de máster. Se plantean cada una de las tareas agrupadas en cinco secciones, correspondientes a cada una de las partes del proyecto. Algunas fechas se solapan entre ellas, ya que se pueden realizar en paralelo por su similitud. La figura 1.1 expone el resumen de las tareas y las fechas.

Resumen de tareas

Tarea

Nombre	Fecha de inicio	Fecha de fin
INICIO DEL PROYECTO	4/4/22	8/4/22
Planteamiento inicial	4/4/22	5/4/22
Preparación del entorno de pruebas y herramientas	6/4/22	8/4/22
Selección de las herramientas de software	6/4/22	8/4/22
Estudio de viabilidad y recopilación de recursos	6/4/22	8/4/22
EJEMPLO DE POSIBLE TÉCNICA DE OCULTACIÓN Y ENVÍO MASIVO DE RANSOMWARE	11/4/22	20/4/22
Documentación sobre técnicas de ocultación de malware	11/4/22	11/4/22
Investigación sobre métodos de ocultación de malware	12/4/22	13/4/22
Documentación sobre Spear-Phishing y EmailSpoofing	18/4/22	18/4/22
Investigación sobre Spear-Phishing	19/4/22	20/4/22
Investigación sobre EmailSpoofing	19/4/22	20/4/22
ANÁLISIS DE RYUK (ESTÁTICO, DINÁMICO, CÓDIGO)	21/4/22	5/5/22
Documentación sobre el Ransomware	21/4/22	22/4/22
Análisis estático	25/4/22	27/4/22
Análisis dinámico	28/4/22	2/5/22
Análisis de código	3/5/22	5/5/22
USO DE LA CRIPTOGRAFÍA EN RYUK Y MEDIDAS DE MITIGACIÓN CONTRA EL CIFRADO DE DATOS	6/5/22	16/5/22
Investigación sobre criptografía en Ryuk	6/5/22	9/5/22
Documentación sobre API hooking	10/5/22	11/5/22
Creación y pruebas herramienta HectorCryptoHooking	12/5/22	16/5/22
USO DE TÉCNICAS DE MACHINE LEARNING CON CARACTERÍSTICAS ESTÁTICAS DEL RANSOMWARE	17/5/22	27/5/22
Documentación sobre aprendizaje automático en Ransomware	17/5/22	18/5/22
Investigación y creación modelos predictivos	19/5/22	27/5/22

Figura 1.1 Resumen de tareas en la planificación.

Como puntos a destacar:

- Una primera fase “Inicio del proyecto” organiza cuáles serán las secciones del conjunto del trabajo a realizar para, después, preparar el entorno de pruebas, seleccionar

herramientas y realizar los estudios pertinentes. Estas tareas guardan relación entre ellas y podrían llevarse a cabo simultáneamente en dos semanas.

- Siempre existe una tarea de documentación antes de realizar investigaciones o análisis que conlleven un trabajo práctico. Se incluiría también adecuar el entorno de pruebas con las herramientas y configuraciones oportunas si no estuvieran ya disponibles.

- Los capítulos sobre análisis estático, dinámico y de código se agrupan en la misma fase, estando el inicio de cada uno de ellos ligado a la fecha final del anterior. No se ha considerado conveniente comenzar el análisis dinámico sin haber finalizado el estático. La duración de todos los análisis será de aproximadamente un mes.

El diagrama de Gantt resultante de la planificación se muestra en la figura 1.2 y el desglose de horas en la tabla 9.1:

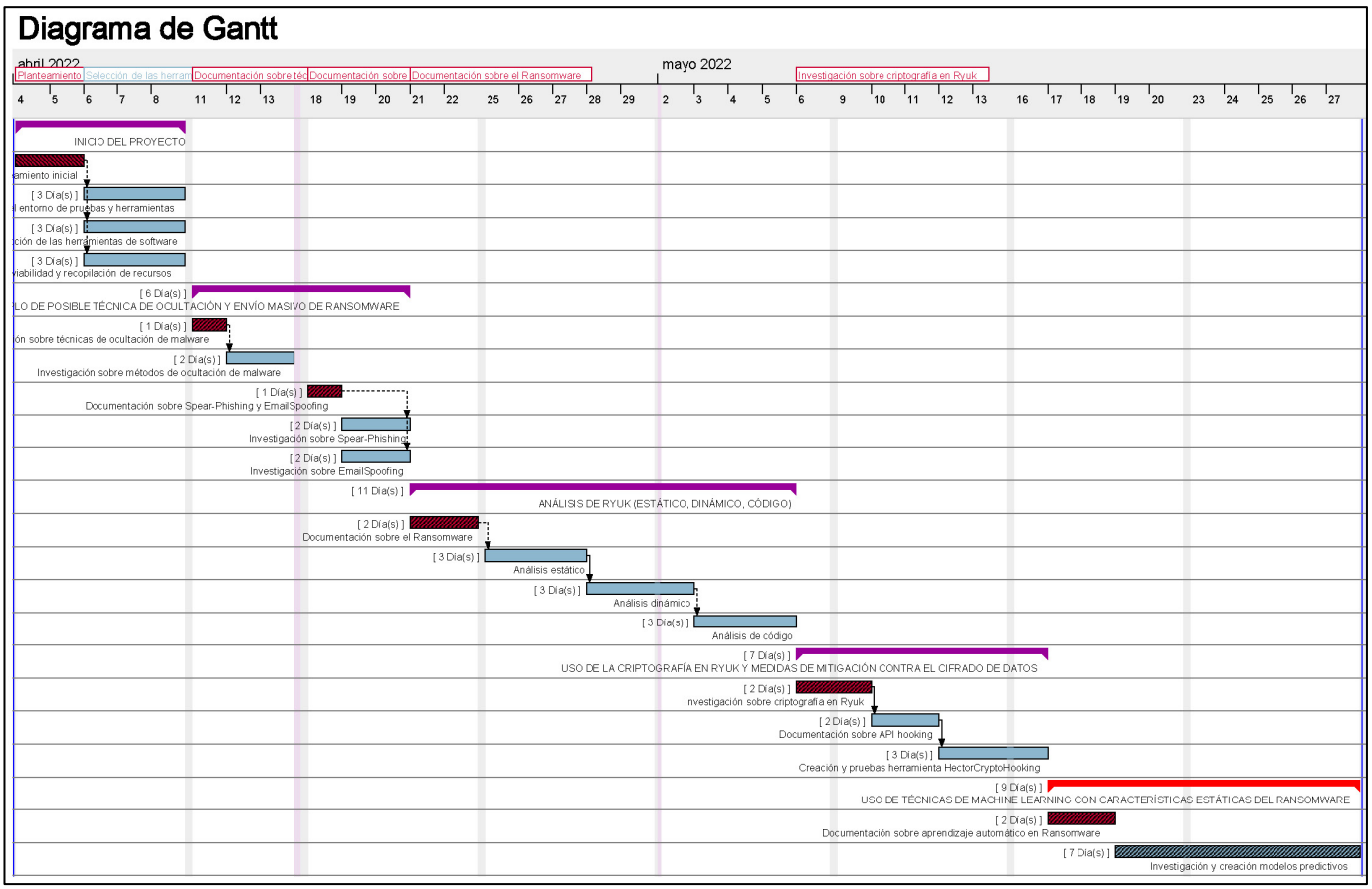


Figura 1.2 Diagrama de Gantt de la planificación.

Descripción de tarea	Horas estimadas	Días (Jornada 8 horas)
Fase de Planteamiento y preparación inicial del proyecto	**40**	**5**
• Planteamiento inicial	16	2
• Preparación del entorno de pruebas y herramientas • Selección de las herramientas de software • Estudio de viabilidad y recopilación de recursos	24	3
Fase de Investigación sobre técnicas de ocultación y envío masivo de ransomware	**48**	**6**
• Documentación sobre técnicas de ocultación de malware	8	1
• Investigación sobre métodos de ocultación de malware	16	2
• Documentación sobre Spear-Phishing y EmailSpoofing	8	1
• Investigación sobre Spear-Phishing • Investigación sobre EmailSpoofing	16	2
Fase de Análisis estático, dinámico y de código para Ryuk	**88**	**11**
• Documentación sobre el Ransomware	16	2
• Análisis estático	24	3
• Análisis dinámico	24	3
• Análisis de código	24	3
Fase de Investigación sobre criptografía en Ryuk y medidas de mitigación en el cifrado de datos	**56**	**7**
• Investigación sobre criptografía en Ryuk	16	2
• Documentación sobre API hooking	16	2
• Creación y pruebas herramienta HectorCryptoHooking	24	3
Fase de Investigación sobre aprendizaje automático para predecir ransomware	**68**	**8,5**
• Documentación sobre aprendizaje automático en Ransomware	16	2
• Investigación y creación modelos predictivos	52	6,5
Total Horas: 300	**Total días: 37,5**	

Tabla 1.1 Desglose de horas por cada tarea del proyecto

El total de días transcurridos desde el inicio al fin del proyecto son 37,5 días laborables en jornadas de 8 horas, que hacen un total de 300. Dentro del desarrollo de este proyecto existen varios roles y personas implicadas:

- Jefe de proyecto: Planificará, organizará y coordinará todas las acciones y eventos. En la fase inicial de planteamiento asumirá casi toda la carga de trabajo. En las demás fases

tendrá una cantidad reducida de horas supervisando y atendiendo posibles problemas e imprevistos que pudieran surgir.

- Técnico de sistemas: Preparará la infraestructura hardware y software involucrada en las distintas fases. Por ejemplo, creación de máquinas virtuales, servidores, dominios, montaje de equipos informáticos, etc.

- Analista de seguridad informática: Posee conocimientos sólidos en ciberseguridad. En la fase de investigación sobre técnicas de ocultación y envío de ransomware trabajará la mayor parte de las horas asignadas. En la fase de investigación sobre criptografía dará soporte al Analista – Programador en los algoritmos criptográficos y el funcionamiento de las APIs que utiliza el ransomware.

- Analista de malware: Examinará y diseccionará el ransomware con análisis estáticos, dinámicos y de código. También estará involucrado en la última fase por su conocimiento sobre la estructura de los ficheros ejecutables y las pruebas en el ransomware.

- Analista programador: Será encargado del análisis y desarrollo del software construido para el API-Hooking.

- Ingeniero de inteligencia artificial: Creará los scripts necesarios para generar los modelos predictivos y ejecutar las pruebas pertinentes. En la última fase del proyecto será el encargado del entrenamiento de los modelos, asegurando su fiabilidad.

Cada uno de estos perfiles tendrá un coste diferente por hora trabajada y lo forma un solo profesional en cada tarea. En la tabla 1.2 se detalla el profesional asignado a cada fase, junto con el número de horas y cantidad en euros. El coste por hora varía según el rol del empleado, resultando el monto total por la mano de obra en 11.940€.

Descripción de tarea	Horas estimadas	Coste tarea(€)
Fase de Planteamiento y preparación inicial del proyecto	**40**	**1.840**
• Jefe de proyecto	32	1.600
• Técnico de sistemas	8	240
Fase de Investigación sobre técnicas de ocultación y envío masivo de ransomware	**48**	**2.000**
• Jefe de proyecto	8	400
• Analista de seguridad informática	40	1.600
Fase de Análisis estático, dinámico y de código para Ryuk	**88**	**3.200**
• Jefe de proyecto	8	400
• Analista de malware	80	2.800
Fase de Investigación sobre criptografía en Ryuk y medidas de mitigación en el cifrado de datos	**56**	**2.130**
• Jefe de proyecto	8	400
• Analista - Programador	38	1.330
• Analista de seguridad informática	10	400
Fase de Investigación sobre aprendizaje automático para predecir ransomware	**68**	**2.770**
• Jefe de proyecto	10	500
• Ingeniero de inteligencia artificial	48	1.920
• Analista de malware	10	350
Precio por hora:		
Jefe de proyecto: 50€	66	3.300
Técnico de sistemas: 30€	8	240
Analista de seguridad informática: 40€	50	2.000
Analista de malware: 35€	90	3.150
Analista – Programador: 35€	38	1.330
Ingeniero de inteligencia artificial: 40€	48	1.920
Total	**300 horas**	**11.940€**

Tabla 1.2 Desglose de costes por tarea y rol.

El coste de los recursos de hardware y software se compone de la computadora utilizada para las pruebas y desarrollos, junto con las licencias de software de los sistemas operativos y paquetes de software propietarios. La tabla 1.3 especifica los recursos empleados con las cantidades en euros. Solamente se incluye el software que haya supuesto un coste económico y se considera que, únicamente, son necesarios dos ordenadores: uno para realizar las tareas de entrenamiento del modelo y virtualizar las instancias de Windows y Ubuntu, formando el entorno de pruebas del malware; otro menos potente para para labores de programación y ofimática. El precio del hardware y software implica un gasto de 4.424,36€

Descripción del recurso	Características	Coste (€)
Computadora para la creación del entorno virtual y entrenamiento de los modelos predictivos	MSI Creator P50 11SI-076ES • Procesador: Intel Core I7-11700 • Memoria RAM: 32GB • Almacenamiento: 1TB+512GB SSD • Sistema Operativo Windows 10 pro 64	1.519
Computadora para labores de programación y desarrollo	ThinkBook 15 G2 ITL • Procesador: Intel Core I5-1135g7 • Memoria RAM: 16GB • Almacenamiento: 512GB SSD • Sistema Operativo Windows 10 Pro 64	899
Memoria USB para almacenar o trasladar información puntualmente	Kingston DataTraveler Exodia 256GB USB 3.2	41,99
Teclado + ratón inalámbrico	Logitech MK295	26
Licencia Windows para máquina virtual 1	Licencia Windows 10 Pro Retail	259
Licencia Windows para máquina virtual 2	Licencia Windows 10 Pro Retail	259
Licencia VMware Workstation Pro para la administración de las máquinas virtuales	VMware Workstation 16 Pro	180,98
Licencia de Microsoft Visual Studio para la compilación de Deviare2 y el desarrollo de la herramienta de API-Hooking	Microsoft Visual Studio 2017 Professional	1.239,39
	Total:	**4.424,36**

Teniendo en cuenta el coste de la mano de obra y de los recursos hardware/software empleados, la cantidad total que deberá ser abonada para este proyecto asciende a 16.364,36€.

2. Fundamentos

2.1- Estado del arte y elección

Según publica Kochovski, A. en Cloudwards (enero de 2022), el coste de los ataques por ransomware en 2021 ascendió a 20.000 millones de dólares viéndose afectadas el 37% de todas las empresas y organizaciones existentes en el mundo. Obviamente se desconocen el número de usuarios particulares afectados por este malware, pero se estima que el 32% termina por pagar el rescate con una tasa de recuperación de datos del 65%.

Dentro del contexto sobre la distribución y ocultación de malware algunos autores, como Yan y Zhang (octubre de 2008) ya exponen el problema y particularidades que supone el empaquetado de malware y Monnappa (2018) describe la forma en la que los recursos de un archivo ejecutable pueden ser alterados. Por otro lado, las amenazas de ransomware se distribuyen como cualquier otro malware, de hecho, según las estadísticas, el 98% de los ciberataques se basan en la ingeniería social y el 92% del malware es enviado por email. (Purplesec, 2021).

En lugar de experimentar con programas de empaquetado populares como UPX o Exe Packer, se optó por emplear una combinación de técnicas al alcance de muchos usuarios en cuanto a recursos y nivel de complejidad se refiere.

El software escogido para las pruebas de distribución del ransomware, Gophish, garantizaba utilizar un framework de código abierto con múltiples opciones de configuración, posibilidad de modificación de cabeceras y una interfaz gráfica intuitiva. Autores como Kumar,V y Beggs,R. (2019), lo recomiendan en auditorios de penetración. Si bien es cierto que existen alternativas a Gophish gratuitas, como Kingphisher, la mayoría de sus funcionalidades no se necesitaban y no estaban tan centradas a los ataques de ingeniería social.

Respecto al análisis sobre el ransomware Ryuk, la decisión ya se había tomado antes de comenzar este proyecto, siendo un ransomware que había cobrado gran relevancia en el último año, especialmente en España, por el ataque sufrido en el SEPE (Servicio Público de Empleo Estatal). Existe abundante documentación sobre él disgregada en informes (Centro

Criptológico Nacional, 2021) y revisiones varias en páginas web especializadas. A pesar del impacto de Ryuk, en el momento de la entrega del este trabajo de fin de máster, cobran importancia otras amenazas ransomware, por ejemplo, la considerada como "sucesor de Ryuk": Conti.

En este trabajo se creyó necesario un análisis más meticuloso y profundo, introduciéndolo en el marco que engloba la distribución, medidas de mitigación ante el cifrado de ficheros y detección temprana a través de machine learning. Se toma en todos esos ámbitos como punto de referencia el funcionamiento de Ryuk.

En referencia al estudio sobre el cifrado de ficheros en Ryuk y formas de mitigación mediante API Hooking, existen referencias de publicaciones en conferencias de ciberseguridad centrada en la utilidad del enganche en las API en el análisis dinámico con las librerías Deviare. (Salah y Marhusin, 2019). Sin embargo, no se encontró documentación referida al API Hooking para la recuperación de claves y el bloqueo del ransomware.

Se tuvieron en cuenta otras librerías con las mismas funciones que Deviare, pero no llegaron a encontrarse tan versátiles y disponibles en tantos lenguajes de programación como esta. Se ha de reconocer que uno de sus inconvenientes es la limitada documentación y falta de información disponible para su implementación y uso.

La aproximación final al aprendizaje automático para la clasificación de binarios tuvo principalmente en cuenta el trabajo de Ashraf, A. y Aziz, A del instituto de Ingeniería y Ciencias Aplicadas de Pakistán, que en su artículo de investigación "Ransomware Analysis using Feature Engineering and Deep Neural Networks" (2019). recopilaron dos datasets, uno compuesto por datos estáticos y otro por datos dinámicos para, finalmente, hacerlo público en el repositorio Github: https://github.com/arxlan786/Malware-Analysis. La relación del TFM con este trabajo únicamente viene dada por la recopilación de características estáticas que hicieron públicas, puesto que el alumno se ha basado en otras publicaciones científicas a la hora de escoger las características sobre las que aplicar los algoritmos de clasificación, pero también en su propio criterio. [1] [2].

2.2- Metodología del trabajo

Es necesario para toda investigación con malware crear un entorno de máquinas virtuales seguro, conectadas entre sí sin posibilidad de establecer conexión con el PC anfitrión. Los elementos y características de esta red virtual son:

- Implementadas bajo VMware Workstation 16 Pro en su versión 16.2.2 build-19200509.

- El host que contiene las máquinas virtuales es un equipo físico con Windows 10 Pro-64-bit versión 21H2 con:

  - 16 GB de memoria RAM (Random Access Memory) a 3200 MHz

  - Microprocesador AMD Ryzen 2700 (8 núcleos, 16 hilos) con velocidad de reloj base a 3,2 GHz y velocidad turbo hasta 4.1 GHz

  - El almacenamiento de las máquinas virtuales en una unidad SSD (Solid State Drive) bajo interfaz SATA III (Serial Advanced Technology Attachment) con velocidades de lectura secuencial de 545 MB/s y escritura secuencial de 435 MB/s

- Máquina virtual Ubuntu versión 20.0.4.3.0 x64 con 2 GB de RAM y 2 núcleos Ryzen 2700 asignados.

- Máquinas virtuales Windows 10 Enterprise 20H2 x64 (versión de evaluación) con 4 GB de RAM y 3 núcleos con 6 hilos AMD Ryzen 2700 asignados.

Se simulan servicios de internet con Inetsim en su versión 1.3.2 de tal modo que existe una conexión en modo Host-Only entre las dos máquinas virtuales con la configuración de red que aparece en la tabla 2.1:

MV. UBUNTU			
Interfaz de red **ens38**		**Host-Only**	
Dirección IP	Máscara de subred	Puerta de enlace	Modo
192.168.10.10	255.255.255.0	192.168.10.1	Estático

Interfaz de red **ens33**	**NAT**
Configurada como DHCP (Desactivada)	

MV. WINDOWS 10			
Adaptador **Ethernet0**		**Host-Only**	
Dirección IP	Máscara de subred	Puerta de enlace	Servidor DNS
192.168.10.21-22	255.255.255.0	192.168.10.10	192.168.10.10
Interfaz de red **Ethernet1**		**NAT**	
Configurada como DHCP (Desactivada)			

Tabla 2.1 Datos de red de ambas máquinas virtuales.

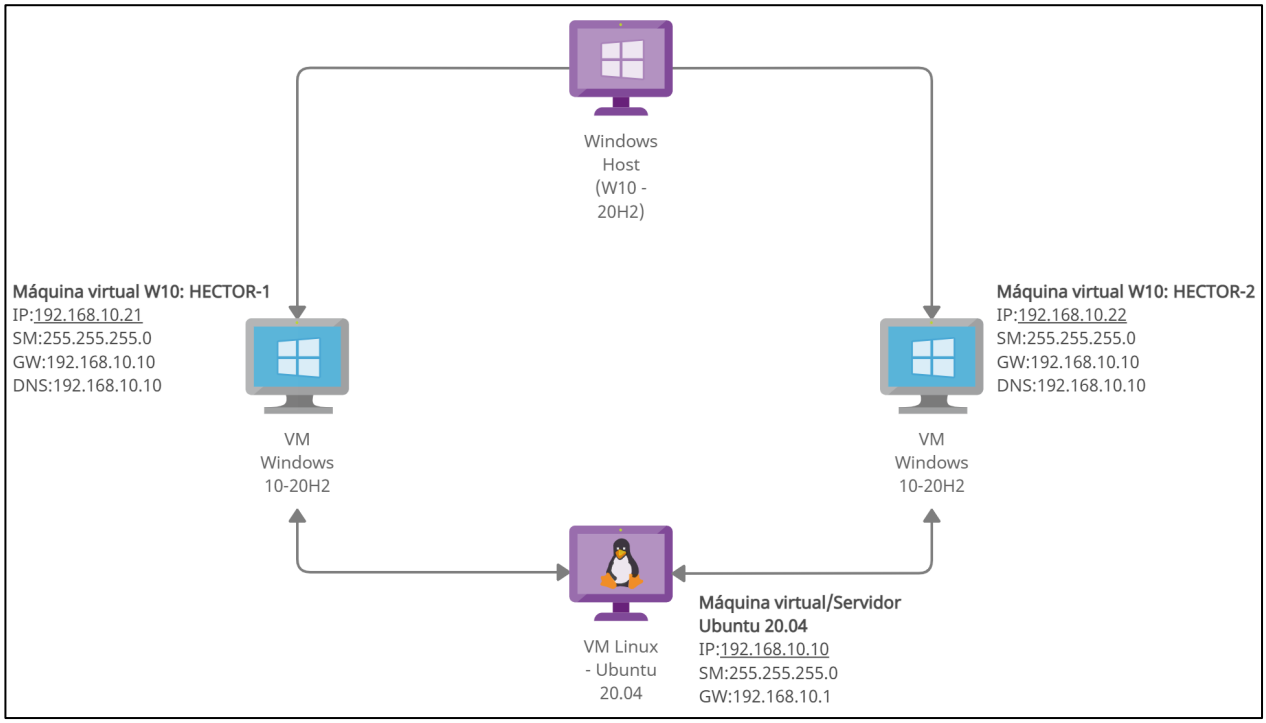


Figura 2.1 Diagrama de red con arquitectura virtual.

La figura 2.1 detalla una computadora física principal: “Windows Host”, cuyo sistema operativo emulará tres máquinas virtuales: “Máquina virtual W10: Hector-1”, “Máquina virtual W10: Hector-2” y “Máquina vitual/Servidor Ubuntu 20.04”.

Estas tres máquinas cuentan con dos interfaces de red, la primera en modo Host-Only a través de la cual estarán conectadas según las direcciones IP (dirección del protocolo de internet que identifica en la red a un dispositivo), máscaras de subred y puerta de enlace, formando una red cerrada entre ellas sin conexión a internet que no interferirá con el sistema

operativo anfitrión; la segunda interfaz comparte la red con el host principal y se encontrará desactivada, reservándola únicamente para descargar actualizaciones o paquetes de software cuando no hay ningún tipo de malware en ejecución.

Todo el tráfico de red de las máquinas virtuales Windows transcurre entre la máquina virtual Ubuntu desde el que se tomarán capturas de este, además de hacer las funciones de servidor de directorio activo para los apartados del análisis dinámico que requieran que los sistemas operativos Windows estén bajo un dominio.

2.3- Software utilizado

2.3.1- Express, Resource Hacker, Hyperion, Gophish

En la primera sección del trabajo realizado sobre ransomware, donde se exponen algunas técnicas para camuflar programas maliciosos como aplicaciones legítimas y las posibilidades de distribución masiva a través de spear-phishing, fueron utilizados las siguientes herramientas:

IExpress (v2.0). Propiedad de Microsoft e incluido en sus sistemas operativos de Windows 2000, genera paquetes de instalación. En uno de estos paquetes, se embebe la muestra de ransomware simulando un documento PDF legítimo. La ejecución de IExpress se efectúa en su interfaz gráfica, desde una máquina virtual Windows 10.

Resource Hacker (v5.1.8). Freeware publicado y mantenido por Angus Johnson, es capaz de editar los archivos de recursos de aplicaciones para Windows 32bit y 64bit. Con él pudieron modificarse los iconos del paquete de instalación IExpress mencionado anteriormente. La ejecución de Resource Hacker se efectúa en su interfaz gráfica, desde una máquina virtual Windows 10.

Hyperion (v2.3.1). Encriptador de archivos PE 32-bits y 64-bits. Su propósito es hacer el contenido original invisible a programas antivirus o de análisis, pues el código se cifra con AES-128 (Advanced Encryption Standard) y se descifra en tiempo de ejecución para que programa incrustado en su interior se ejecute. Se realizan pruebas de cifrado en el paquete de

instalación con Ryuk, para comprobar si pudiera ser detectado por otros antivirus. La ejecución de Hyperion se efectúa por consola de comandos, desde la máquina virtual Ubuntu, junto con MinGW.

VirusTotal. Servicio web que analiza archivos y documentos de todo tipo en busca de software malicioso empleando múltiples programas HIDS/HIPS. Genera automáticamente los hashes e informa de la familia de malware a la que supuestamente podría pertenecer, según cada marca de antivirus.

Gophish (v0.11.0). Herramienta de código abierto multiplataforma para la "simulación" de ataques phishing. Provee funciones para la conexión a servidores de correo y modificación de cabeceras de correo. Se simularon ataques spear-phishing con servicios de correo gratuitos y un servidor local. La ejecución de Gophish se efectúa en su interfaz gráfica, desde la máquina virtual Ubuntu.

2.3.2- Detect it Easy, PEstudio-CLI, PE-bear, BinText, HxD

En la información obtenida del análisis estático, se han aprovechado las funcionalidades de estas herramientas gratuitas.

HxD (v2.5.0.0) desarrollado por Maël Hörz, es un editor hexadecimal para sistemas operativos Windows, rápido, con opciones de búsqueda y funciones que permiten trabajar con el código. Tanto el ransomware como los archivos cifrados se han examinado en este editor para comprobar la existencia de secciones o encontrar ciertos caracteres ASCII que pudieran ser de interés. La ejecución de HxD se efectúa en su interfaz gráfica, desde una máquina virtual Windows 10.

BinText (v3.0.3) propiedad de McAfee, Inc, es un buscador de cadenas de caracteres ASCII o UNICODE para Windows con modalidades de filtrado por tipo de carácter, tamaño, número de caracteres consecutivos u obligatorios, de gran utilidad en el momento de encontrar evidencias sobre el comportamiento del malware según los nombres de las variables, direcciones IP, funciones, llamadas a API, comentarios y cualquier texto relevante. La

ejecución de BinText se efectúa en su interfaz gráfica, desde una máquina virtual Windows 10.

PE-bear (v0.5.5.4) se trata de un proyecto de software libre desarrollado por HaSherezade, para sistemas Windows y Linux, que aglutina un completo conjunto de herramientas y funciones de análisis para archivos PE (Portable Executable). Ha proporcionado datos sobre la estructura interna de Ryuk en cuanto a secciones, cabeceras y tablas. La ejecución de PE-bear se efectúa en su interfaz gráfica, desde una máquina virtual Windows 10.

PEstudio-CLI (v9.39) creado por Marc Ochsenmeier, en su versión gratuita, ha proporcionado información complementaria a PE-bear, pues ambos sirven para el mismo propósito. Es capaz de clasificar por tipo ciertas cadenas de caracteres incrustadas en el código, lo cual, se cotejó con el listado de BinText agilizando el proceso de búsqueda. La ejecución de PEstudio se efectúa en su interfaz gráfica y en consola de comandos, desde una máquina virtual Windows 10.

Detect It Easy (v3.04). Este programa bajo licencia MIT puede detectar determinados atributos de un archivo binario que indican si está empaquetado, encriptado o cómo se encuentra compilado, además de indicar la forma de revertir el proceso según el packer o crypter correspondiente. La muestra de Ryuk se analizó con la interfaz gráfica con el fin de detectar posibles empaquetados y calcular el nivel de entropía. La ejecución de Detect It Easy se efectúa en su interfaz gráfica, desde una máquina virtual Windows 10.

2.3.3- Process Hacker, Autoruns, Wireshark, Sysmon, Procmon y Noriben

El análisis dinámico de Ryuk, principalmente, tuvo en cuenta estas herramientas de uso generalizado para este propósito.

Process Hacker (v2.39) es un programa *open source*, diseñado para depurar software y detectar malware. Presenta un monitor de recursos avanzado con información valiosa sobre los procesos del sistema: llamadas a DLLs (dynamic-link library), conexiones de red, gráficos de actividad, posibilidad de realizar volcados de memoria. Además, cuenta con la ventaja de ser capaz de detener cualquier proceso. En el análisis dinámico expuesto en este trabajo,

además de revelar interacciones con otros procesos, la memoria y recursos del sistema, fue útil para detener el proceso malicioso sin restricciones de ningún tipo. La ejecución de Process Hacker se efectúa en su interfaz gráfica, desde las máquinas virtuales Windows 10.

Autoruns (v14.09) de Mark Russinovich, indica las entradas añadidas en diferentes secciones de los sistemas operativos Windows de los programas instalados, para automatizar su arranque automático al inicio del sistema. Por otro lado, también pueden verse recursos del sistema asociados a estas aplicaciones. Puesto que es probable que el ransomware asegure su supervivencia frente al reinicio de la computadora, es una aplicación imprescindible. La ejecución de Autoruns se efectúa en su interfaz gráfica, desde las máquinas virtuales Windows 10.

Sysmon (v14.0) para Windows, creado por Mark Russinovich y Thomas Garnier, proporciona supervisión y registro de eventos del sistema avanzados. Permite cargar archivos de configuración personalizados y la recopilación actividad de procesos, conexiones y cambios en el sistema de archivos es muy detallada

Procmon (v3.91) y **Noriben**. El primero es un monitor de procesos gratuito creado por Mark Russinovich, capaz de capturar todas o la mayoría de las interacciones de un proceso con el sistema operativo. Las capturas son exportables para un posterior análisis. Esta herramienta es fundamental en el análisis dinámico realizado, habiendo mostrado muchos de los principales comportamientos de Ryuk. Por su lado, Noriben es un conjunto de scripts escritos en Python que funciona con Procmon cuya particularidad es que puede detectar procesos maliciosos monitorizándolos solo a ellos y reduciendo el ruido de los que no representan amenazas. La ejecución de Procmon y Noriben se efectúa en su interfaz gráfica y en PowerShell, desde las máquinas virtuales Windows 10, junto con Python.

Wireshark (v2.4.9). El más extendido analizador de protocolo de red. En su versión para Linux, tanto por consola (TShark) para capturar tramas, como con su interfaz gráfica, se monitorizó y analizó el tráfico producido en los intentos de pivotar de Ryuk.
La ejecución de Wireshark se efectúa en su interfaz gráfica y en la consola de comandos, desde la máquina virtual Ubuntu.

2.3.4- X64dbg, ScyllaHide

Es posible que solo los desarrolladores de Ryuk tengan acceso a su código fuente escrito C/C++, por lo que para examinar su estructura se debe utilizar un depurador o desensamblador. Estas herramientas muestran el código en el lenguaje de programación de bajo nivel ensamblador y son capaces de interactuar con el binario ejecutando instrucción por instrucción, mientras ofrecen visualizar el contenido de la memoria del sistema y los registros del microprocesador. Son de gran utilidad para indagar en profundidad en el funcionamiento del ransomware.

X64dbg (v2022-09-01_00). Depurador de nivel ensamblador con el que se puede realizar un proceso de ingeniería inversa de cualquier binario, tanto en su versión para 32 bits (x32dbg) como en la versión de 64 bits (x64dbg). Muestra el código ensamblador, diagramas de flujo con la secuencia de ejecución de las funciones e inspector de llamadas entre módulos y cadenas de caracteres entre otras funciones. Admite gran variedad de plugins. Fue utilizado para el análisis de código de Ryuk, ejecutando paso a paso sus instrucciones o estableciendo puntos de interrupción en las llamadas a APIs de Windows.

ScyllaHide. Librería especializada en evadir sistemas anti-depuración. Se integra en X64dbg como un plugin. Su uso no fue estrictamente necesario, aunque sí solucionó algunos problemas de inestabilidad en el sistema operativo en momentos aleatorios de la depuración.

2.3.5- Ubuntu Server, Samba Active Directory, Kerberos y Winbind

Para experimentar con el movimiento lateral del ransomware analizado, tuvo que generarse un dominio con dos usuarios para cada máquina virtual Windows. Requirió de la instalación de varias herramientas que se detallan en este punto.

La decisión de utilizar software libre, en esto caso soluciones Linux, fue motivada por el escaso consumo de recursos del sistema que les caracteriza en comparación con los productos de Microsoft; virtualizar tres sistemas operativos al mismo tiempo requería de

una cantidad de memoria RAM mayor a la que el alumno poseía. Por otro lado, de este modo se le da un enfoque diferente al tradicional sistema de directorio activo para dominios en sistemas operativos Windows comúnmente utilizados, normalmente distintas versiones de Windows Server.

Ubuntu Server (v20.04). Distribución más utilizada en servidores Linux. La máquina con Linux dentro del entorno virtual lleva esta distribución sin la interfaz gráfica con el fin de crear un dominio para las máquinas virtuales Windows.

Samba Active Directory (v4.3). Se trata de software libre que proporciona a un servidor las funciones para compartir archivos, impresoras y roles de directorio activo. Con el directorio activo los usuarios del directorio se conectan a un dominio y comparten recursos de la red de trabajo.

Kerberos (v5-1.16). Es un servicio de autenticación que, en este contexto, facilitará la autenticación en el dominio y servicio de directorio activo creado. Lo usuarios de las máquinas Windows (Clientes) podrán autenticarse en el servidor Ubuntu mediante este protocolo.

Winbind. Posibilita que una plataforma UNIX entienda la información concerniente a los usuarios y grupos que se intercambia con entornos Windows y viceversa. Las comunicaciones entre Samba Active Directory y las máquinas virtuales Windows 10 no podría funcionar sin el componente Winbind.

2.3.6- Visual Studio, Deviare2

La creación de la herramienta de mitigación ante el cifrado de archivos, *HectorCryptoHooking,* se desarrolló en C# mediante una versión de prueba de Microsoft Visual Studio Enterprise 2017. También este fue necesario para la compilación de las librerías Deviare2 con los siguientes componentes instalados:

- SDK de Windows 8.1
- SDK de .NET Framework 4.6.1
- VC++ 2017 versión 15.9 v14.16

- Runtime de Visual C++ para UWP
- Conjunto de herramientas de VC++ 2017

El entorno fue escogido porque proporcionaba la máxima compatibilidad con la última versión de Deviare.

Las librerías Deviare2, publicadas bajo licencia GPL-3.0, pertenecen a Nektra. En su versión 2.8.3 son útiles en arquitecturas de 32 bits y 64 bits. Muchas compañías hacen usa de esta API en diversos campos de la ingeniería informática. La ejecución de Visual Studio con Deviare2 se efectúa, desde el host Windows 10 y Las pruebas realizadas desde una máquina virtual Windows.

2.3.7- Scikit-Learn

Las pruebas con el set de datos de características estáticas de ransomware, junto con el entrenamiento y creación de los modelos predictivos, se basaron en el desarrollo de scripts en Python implementando los algoritmos de la librería Scikit-Learn. Scikit-Learn es software libre y permite utilizar algoritmos de regresión logística, agrupamiento o clasificación. Tanto XGBoost como Random Forest tienen soporte en esta librería, además de compatibilidad con herramientas que trabajan con matrices, por ejemplo, NumPy.

2.3.8- Otras herramientas

Microsoft Outlook (16.0.11). Perteneciente a la suite Office 365 cuya licencia es proporcionada por la Universidad Nacional de Educación a Distancia, Outlook es una aplicación de gestión de correo con la que se pudo establecer una conexión a un servidor SMTP simulado en la red local.

Notepad++ (8.4). Editor de texto y de código con soporte para Windows y Linux con características avanzadas. Se abrieron algunos archivos de texto con el volcado de las cadenas de caracteres del extraídas de Ryuk, al igual que otros con scripts en Python o comandos batch.

3. Posibles técnicas de ocultación y envío masivo de ransomware

Lo que se va a exponer en esta sección podría considerarse como un conjunto de métodos para engañar y confundir al usuario. Este proceso sigue un conjunto de pasos inverso al que ha de realizar el ciberdelincuente para preparar la amenaza y distribuirla

En este orden, primero se debe engañar al usuario en lo correspondiente a la distribución del programa malicioso. El correo electrónico, un método masivo e idóneo para esta tarea, requiere de un texto elaborado, a ser posible sobre un tema de actualidad o que despierte el interés o inquietud al destinatario. El ejemplo que se va a exponer sobre la factura de la luz puede ser perfectamente válido. La herramienta Gophish empleada para este propósito, se trata de un programa de "simulación" totalmente funcional capaz de enviar emails maliciosos con remitentes e información falsa.

El segundo paso es el de evadir el software de seguridad (si lo hubiera) instalado en la computadora. Se puede hacer uso de un software que encripte el código malicioso como, por ejemplo, Hyperion. Aunque este no es cien por cien eficaz al ser detectado ya directamente por muchos productos antivirus, independientemente del contenido cifrado que albergue; los desarrolladores de malware suelen emplear sus propias técnicas de ofuscación y codificación como en Agent Tesla, Cobalt Strike, DarkHotel… (MITRE|ATT&CK, s.f) para que, temporalmente, sean invisibles a los sistemas de detección de amenazas.

El tercer y último paso consiste en engañar nuevamente a la víctima haciéndole creer que su descarga es un documento o programa legítimo. Las herramientas y recursos capaces de hacerlo se detallarán a continuación.

3.1- Empaquetado, Modificación y cifrado de un fichero Portable Ejecutable

Un recurso útil y simple para empaquetar y ejecutar un programa o código, es el programa IExpress. Este creador de paquetes autoextraíbles puede ser aprovechado con fines maliciosos. En este supuesto se pretende camuflarlo como una factura falsa y para esta tarea existen algunos recursos gratuitos como la web invoicehome.com, que genera facturas a través de plantillas. Se le expondrá a la víctima que la compañía Iberdrola le reclama un cargo extra en su recibo de la luz que, en cierto modo, no resulta descabellado. El procedimiento para generar el paquete autoextraíble consiste en:

- Seleccionar *Extraer archivos y ejecutar comandos de instalación*. Esto posibilitará ejecutar un programa de instalación después de la extracción.

- No incluir ninguna ventana emergente de confirmación ni de acuerdo de licencia ya que el objetivo es que se realice la operación de forma silenciosa.

- Incluir el archivo ejecutable con el malware, la fotografía con la factura y un *.BAT* (archivo de proceso por lotes).

- Ocultar la ventana del programa de instalación, también la animación de la extracción y seleccionar como programa de instalación el archivo .BAT mediante el comando: `cmd /c <nombre de fichero>.bat`

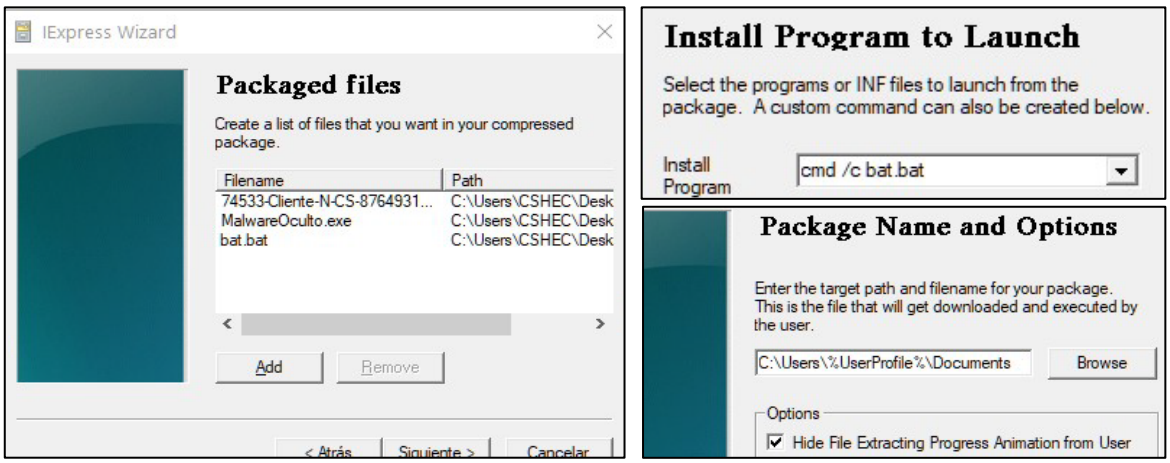


Figura 3.1 Detalles del proceso de creación de un paquete autoextraíble con IExpress.

La figura 3.1 representa los detalles de la creación del paquete y la referencia al archivo .BAT. El contenido de este archivo .BAT contiene dos comandos que ejecutan una instrucción *powershell* para mostrar la factura falsa y arrancar el malware:

```
powershell -WindowStyle Hidden -c "./74533_Cliente_N_CS_87649311.jpg"
powershell -WindowStyle Hidden -c "./RansomwareOcultoRyuk.exe"
```

Habiendo llevado a cabo los pasos anteriores, al ejecutar el paquete generado con IExpress, de forma silenciosa, inicia el ransomware y abre la fotografía con la factura aparentando hacia el usuario que nada está pasando en su computadora.

De cualquier modo, es habitual entre los creadores de malware modificarlo el icono del archivo para aumentar su credibilidad. Esto se consigue fácilmente con *Resource Hacker* para Windows o *PE-bear* para Linux. Como queda plasmado en la figura 3.2, el funcionamiento del software es sencillo y basta con importar un archivo *.ico* (archivo que representa iconos como imágenes) para sustituir el anterior. La imagen escogida es la propia fotografía de la factura falsa que se mostrará. El objetivo siempre será transmitir apariencia de tratarse de un documento auténtico y legítimo.

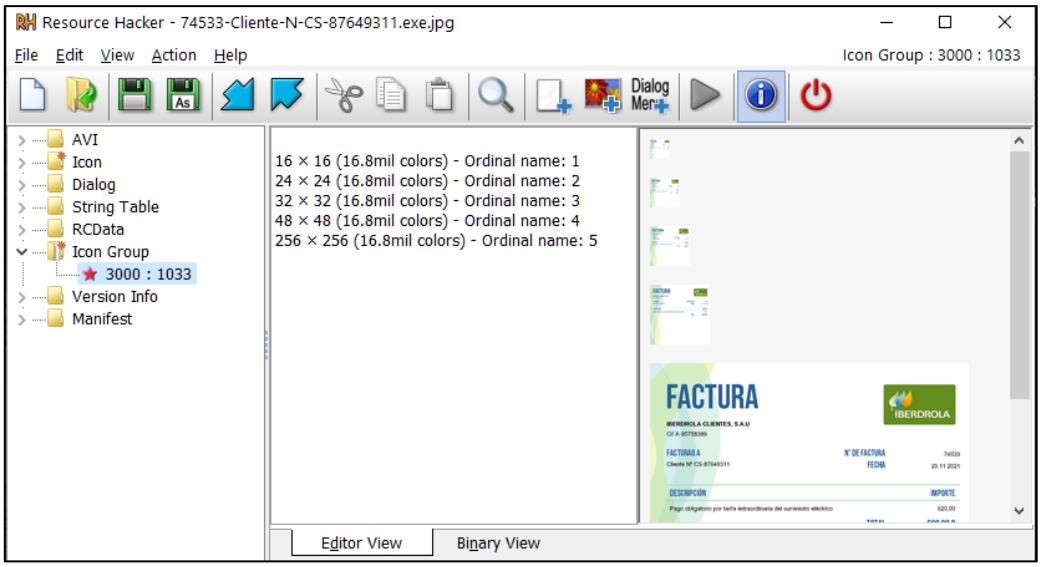


Figura 3.2 Modificando el icono del malware con Resource Hacker.

Yendo más lejos a la hora de camuflar el malware, es posible utilizar el siguiente método con el fin de que la víctima se sienta más confiada y prevenir que descubra el engaño, si tiene habilitado en opciones de carpeta: *"Mostrar extensiones de nombres de archivo"*.

Se trata de insertar un carácter Unicode llamado *right to left override* (RTLO) en el nombre del fichero. Este carácter especial se usa en la escritura y lectura de texto arábico y hebreo e indica que el sistema operativo debe mostrar el texto en el orden contrario al habitual, de derecha a izquierda. De esta forma, el nombre del archivo camuflado pasa de ser *74533-Cliente-N-CS-87649311.jpg.exe* a *74533-Cliente-N-CS-87649311exe.jpg* sin que ello altere la forma de iniciar el ejecutable malicioso. Este Unicode se corresponde con el código **U+202e** y en el siguiente ejemplo se pueden comprobar los resultados de la modificación:

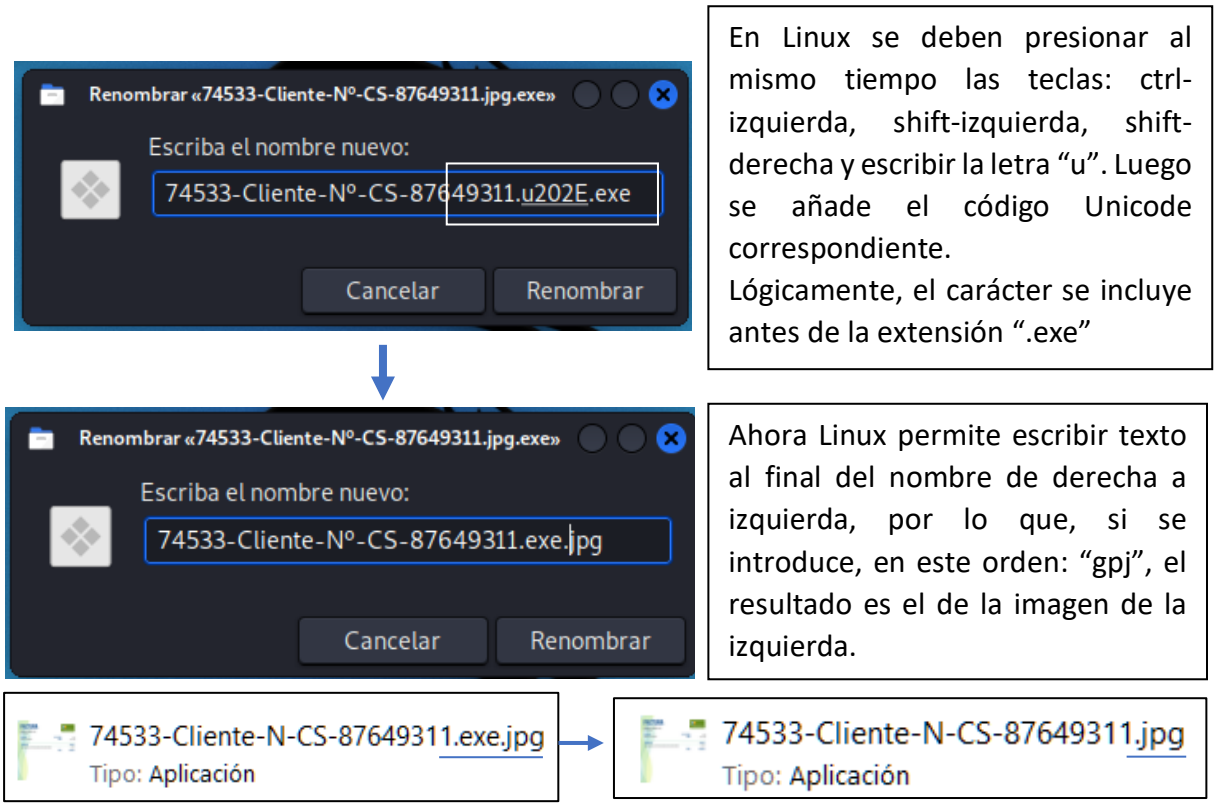


Figura 3.3 Renombrar un fichero utilizando el carácter Unicode RTLO.

La figura 3.3 indica como las modificaciones han tenido éxito al ocultar las extensiones de nombre de archivo e incrustar una falsa vista previa de la factura como icono. La víctima lo

identificará como un fichero de imagen *jpg* (archivo de imagen fotográfica Joint Photographic Experts Group).

En la figura 3.4, se sustituye la factura en forma de jpg por un documento PDF (Portable Document Format). Cuando el supuesto documento es abierto, los comandos introducidos en *bat.bat* provocan la apertura de archivo PDF, pero también ejecuta el ransomware empaquetado en su interior.

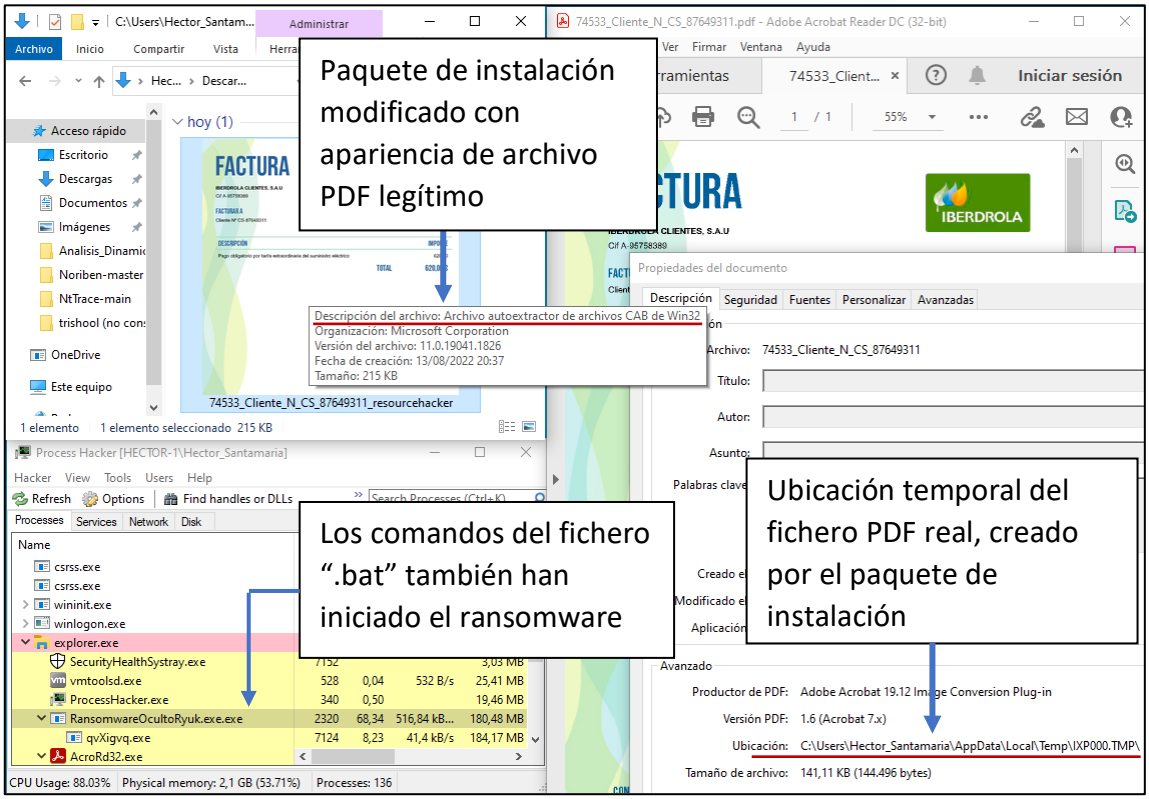


Figura 3.4 Efecto real de apertura del paquete autoextraíble camuflado como documento.

La detección de archivos maliciosos por parte de un software antivirus o sistemas IDS/IPS, puede evitarse camuflando el ransomware o cualquier otro tipo de malware encriptándolo con algún algoritmo de cifrado. Desde los más simples como el cifrado XOR a otros más sofisticados como AES. Una herramienta que permite cifrar un archivo PE con una encriptación AES-128 es Hyperion.

El código fuente de Hyperion debe compilarse con un compilador C como MinGW (Minimalist GNU for Windows) para construir el archivo ejecutable:

```
i686-w64-mingw32-gcc -I Src/Payloads/Aes/c Src/Crypter/*.c
Src/Payloads/Aes/c/*.c -o hyperion.exe
```

Esto genera *hyperion.exe* que, posteriormente, puede usarse mediante la línea de comandos con el ransomware camuflado en la factura falsa. Con el parámetro *-k* se le indica la longitud en bytes de la clave AES aleatoria:

```
hyperion.exe -k 16 -l --verbose "74533-Cliente-N-CS-87649311.exe"
FacturaEncriptada.exe
```

Al finalizar, se construye un nuevo archivo ejecutable (*FacturaEncriptada.exe*) que contiene el binario original cifrado. Al ser FacturaEncriptada.exe ejecutada, procede al descifrado en tiempo de ejecución del original para lanzar el contenido. El esquema que representa el proceso seguido por este y otros crypters similares está en la figura 3.5.

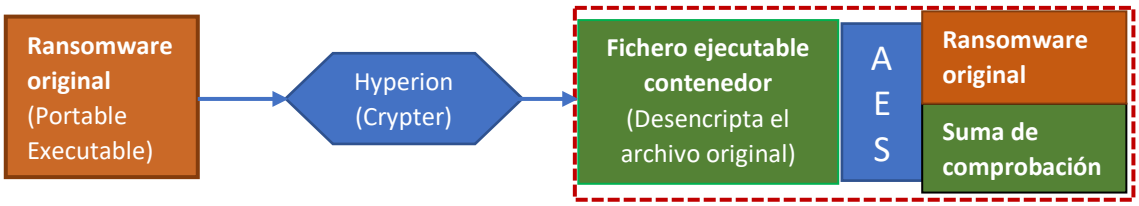


Figura 3.5 Diagrama con la metodología de un crypter como Hyperion. (Elaboración propia).

Este tipo de técnicas evitan que el software malicioso sea detectado, no obstante, el contenedor generado por Hyperion posee unas características que lo hacen detectable por muchos programas antivirus, ya que se trata de software libre bien conocido y disponible para cualquiera que desee llevar a cabo este tipo de modificación.

Un análisis del archivo cifrado creado anteriormente en Virustotal.com, reveló varios positivos en forma de “Malware”, “Unsafe”, “Malicious” y “Trojan”, por soluciones antivirus como ESET-Nod32, Sophos, Windows Defender, Avira y la versión GW de McAfee. Por otro lado, muchos otros programas populares como BitDefender, Avast, Kaspersky, Malwarebytes, Panda y la versión estándar de McAfee, no detectaron ninguna anomalía con el binario del crypter.

Utilizando una herramienta como PPEE se puede obtener un informe del archivo cifrado, cuyos datos se transmitirán automáticamente a VirusTotal para mostrar las coincidencias y el tipo de malware correspondiente. El contenido del informe queda reflejado en la figura 3.6:

```
> Report by Professional PE Explorer (PPEE) - puppy,
https://mzrst.com <

File Size:          267.00 KiB (273408  bytes)
File Entropy:       7.97515
File Type:          64-bit Windows-based
Attributes:         Archive
Created:            01/07/2022  19:12:04:134
Modified:           01/07/2022  19:12:04:136
Accessed:           01/19/2022  18:47:21:787
SSDEEP:
      6144:JGqEsjQ+Ln9wQ6s2GMVb3FyJzdFIIGVO+5Y1x5DU:7EV+5wfb0IIGVO+K
5D
CRC32:              F9D2C810
ImpHash:            A1636159062256538CF7ABFE075BA31F
MD5:          E702DEA846A79220DC0D87453B9144B4
SHA1:         8C8F88BDD196CFC9EB0AA4501ABACB21C08299A1
SHA256:
52A45575FD5F64B1717477A958D7094B165307A299E330CACCF862D5ED8764FB
Authentihash(PE256):9B28BA67B641F8A517118529F2870AFF16E329768A71D9DC
6C0CF737B531C651
```

Figura 3.6 Resumen de características y funciones hash del ransomware encriptado mostrado por la herramienta PPEE.

Antivirus	Detect	Version	Result	Update
Avira	Yes	8.3.3.12	HEUR/AGEN.1136114	20220107
ESET-NOD32	Yes	24583	a variant of MSIL/Kryptik.YNR	20220107
F-Secure	Yes	12.0.86.52	Heuristic.HEUR/AGEN.1136114	20220107
Fortinet	Yes	6.2.142.0	W64/CoinMiner.EJER!tr	20220107
McAfee	Yes	v2019.1.2+3728	BehavesLike.Win64.Generic.dc	20220107
Microsoft	Yes	1.1.18800.4	Trojan:Win32/Sabsik.FL.B!ml	20220107
Sophos	No	1.4.1.0	ML/PE-A	20220107
Acronis	No	1.1.1.82	null	20210512
Avast	No	21.1.5827.0	null	20220107
BitDefender	No	7.2	null	20220107
Kaspersky	No	21.0.1.45	null	20220107
Malwarebytes	No	4.2.2.27	null	20220107
McAfee	No	6.0.6.653	null	20220107
Panda	No	4.6.4.2	null	20220107
……				

Tabla 3.1 Coincidencias por los distintos antivirus del ransomware Ryuk bajo el Crypter "Hyperion".

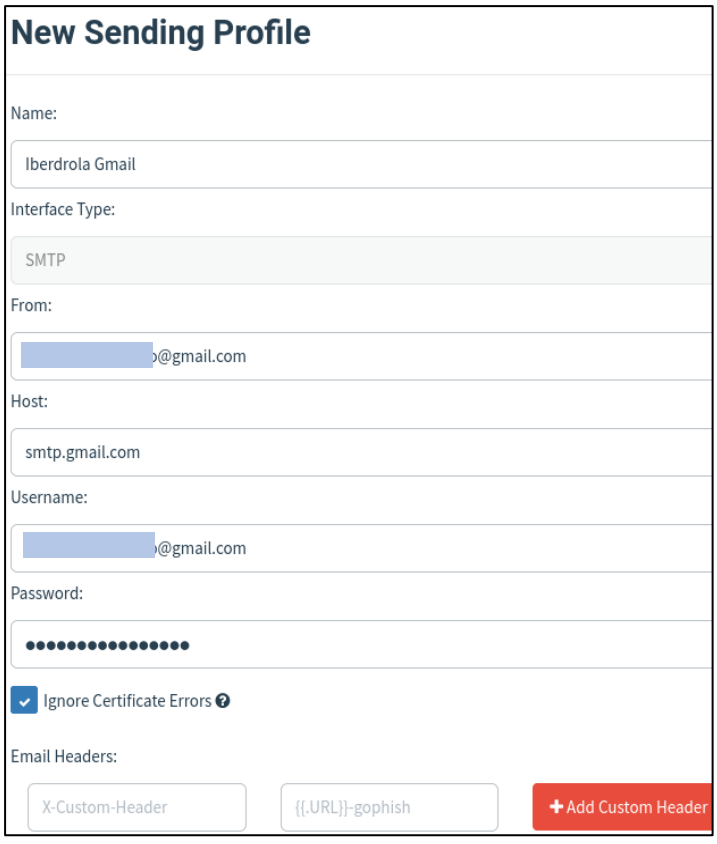
La tabla 3.1 expone parte del total de 19 resultados positivos y 49 resultados negativos para los antivirus consultados en VirusTotal.

Ninguno de ellos detectó la presencia de Ryuk dentro del paquete de instalación generado anteriormente, a su vez dentro del ejecutable producido por Hyperion, ya que ese contenido está encriptado y, aunque se tienen en cuenta las cabeceras y secciones del binario, no pueden verse las del ransomware.

3.2- Envío masivo de ransomware mediante Spearphishing

Un ejemplo válido de cómo podría distribuirse el malware con el que se trabaja en este TFM, consistiría en llevar a cabo una campaña de Spear-Phishing con el fin de que las actividades ilícitas de los ciberdelincuentes llegasen a muchas más personas. Existe un software gratuito bajo licencia Open Source llamado GoPhish que, con fines educativos y de investigación, fue desarrollado para lanzar campañas de phishing. Instalarlo y utilizarlo es relativamente sencillo y muestra de ello son las pruebas realizadas que se detallan a continuación:

El primer paso es conseguir un servidor SMPT desde el que enviar el correo electrónico. Se puede implementar un servidor en Linux con Postfix y hacerse con un nombre de dominio gratuito, pero basta con crear una cuenta de Gmail e introducir los datos y credenciales creando un nuevo perfil en *Sending Profile* de Gophish como muestra la figura 3.7



Gophish permite especificar la dirección de correo desde la que se van a enviar los emails:

correofalso@gmail.com

Son importanes las credenciales para el servidor SMTP (Simple Mail Transfer Protocol) que corresponderían con el servidor por defecto de Gmail y las credenciales del nuevo correo creado Adhoc para la campaña de Spear-Phishing:

smtp.gmail.com
correofalso@gmail.com

En la herramienta se pueden incluir cabeceras personalizadas

Figura 3.7 Creando un perfil de envío nuevo en Gophish

En una situación ideal, donde el atacante poseyera su propio servidor y dominio, es posible hacer *Email Spoofing* al modificar las cabeceras del mensaje y conseguir que el destinatario crea que el remitente es realmente Iberdrola, tal y como se visualiza en la figura 3.8:

Figura 3.8 Cabeceras personalizadas de Gophish para Email Spoofing.

Con servidores SMTP de Google y Microsoft las cabeceras no se vieron modificadas en los envíos y pruebas posteriores, además, esta última compañía directamente no permite el uso de la modificación de la cabecera "**`From:`** `<correo electrónico falso>`" lanzando una excepción durante el proceso. Lo mismo ocurre en el envío de archivos ejecutables desde ambas plataformas, está totalmente restringido.

El segundo paso consiste en cargar una lista de destinatarios que puede hacerse con un archivo CSV. La tabla 3.2 muestra un ejemplo de cómo se estructura ese archivo, haciendo acopio de uno de los múltiples generadores de nombres, apellidos y correos electrónicos aleatorios existentes en internet:

First Name	Last Name	Email	Position
María Isabel	Fernandez	funk.kaela@schneider.net	Cliente
Joan	Garrido	jesse70@leannon.com	Cliente
Victor	Esteban	casper.mavis@marvin.com	Cliente
Héctor	Santamaría	hector@alumno.uned.es	Cliente

Tabla 3.2 Tabla con formato adecuado para importar nombres y correos a Gophish.

El tercer paso implica crear una plantilla que no es más que el correo electrónico que se va a enviar que aparece reflejado en la figura 3.9:

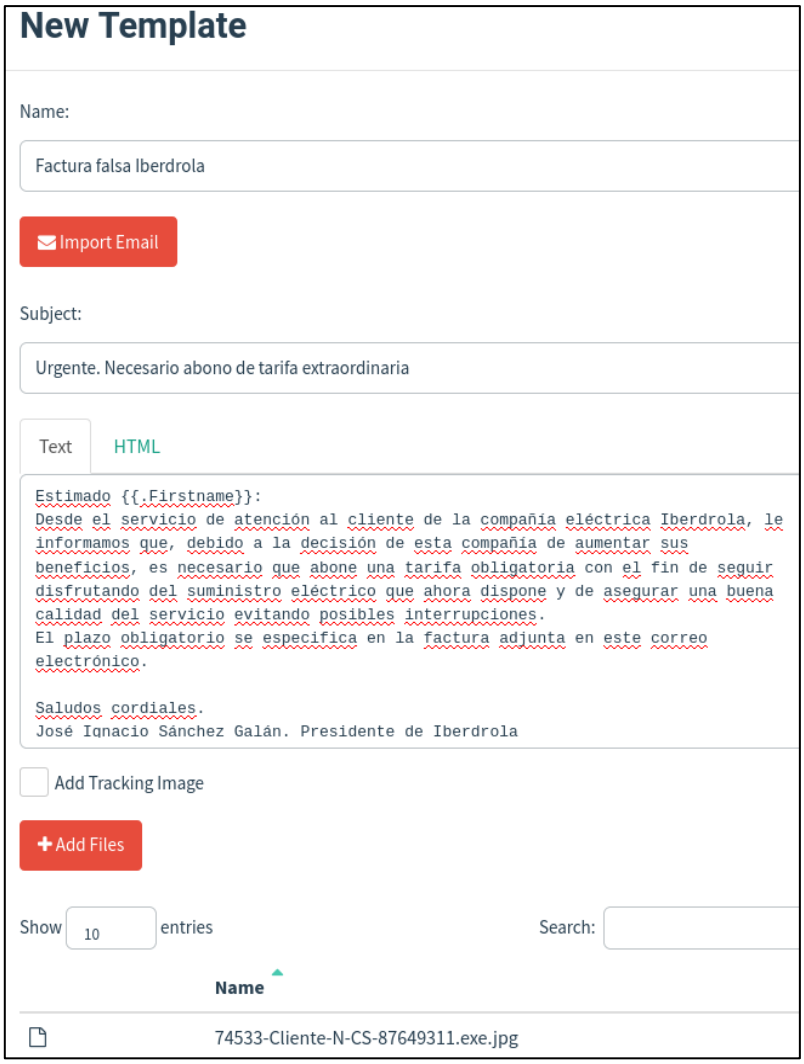


Figura 3.9 Plantilla del correo electrónico que se enviará.

Básicamente se pone a prueba la técnica denominada “Pretexting”, donde se actúa con una identidad falsa. En este caso se utiliza una figura de autoridad incluyendo el nombre del presidente de Iberdrola, pretendiendo manipular al destinatario con el miedo a que le sea interrumpido su suministro eléctrico aprovechando la actual crisis energética y el elevado precio de la luz. Posteriormente se le incita a que abra el documento adjunto sirviéndose el falso correo de su preocupación por la expiración de los supuestos plazos para abonar la factura.

Esta herramienta posibilita insertar varios archivos adjuntos y código JavaScript/HTML así como referencias: `{{.FirstName}}` para personalizar cada email enviado.

- El último paso consiste finalmente en crear una campaña seleccionando perfil de remitente, plantilla y grupos de usuarios cargados.

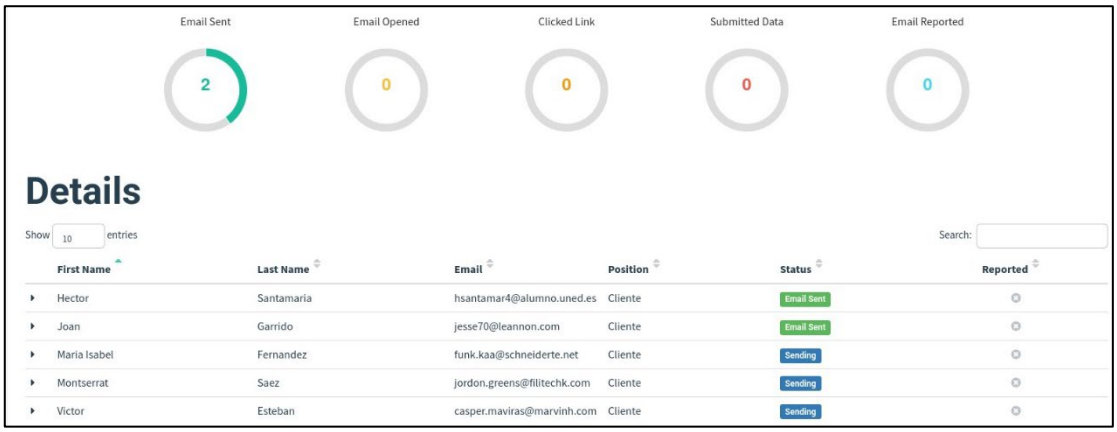


Figura 3.10 Panel de control de los correos enviados donde se recopilan múltiples estadísticas.

El resultado es lo mostrado en la figura 3.10, aunque, al no ser el objetivo de este trabajo de fin de máster, no se profundiza o explotan todas las capacidades de Gophish (ni de lo que al phishing se refiere) que, por ejemplo, permite indicar una dirección URL desde la que la herramienta recibiría información de seguimiento en los emails o el resultado de la introducción de credenciales en una réplica exacta de otra página web, pues este software puede clonar la apariencia visual de los sitios web.

De cualquier modo, la herramienta puede realizar campañas de Spear-Phishing y propagar ransomware o cualquier otro malware que se quiera. Con servidor SMTP y dominio propios, el efecto de modificar las cabeceras por el atacante para engañar al receptor del correo es visible y, dependiendo de las políticas de seguridad de una posible empresa a que vayan dirigidos los correos, es factible que puedan recibir un archivo ejecutable como adjunto.

En las siguientes figuras se muestra como sí se han modificado las cabeceras el email, aunque esto no tenga efecto debido a la seguridad implementada en los servidores SMTP de Google que terminan mostrando el remitente real. El email consigue llegar a su destino tal y como corroboran las capturas de pantalla de las figuras 3.11 y 3.12:

Figura 3.11 Cabeceras modificadas del email spear-phishing.

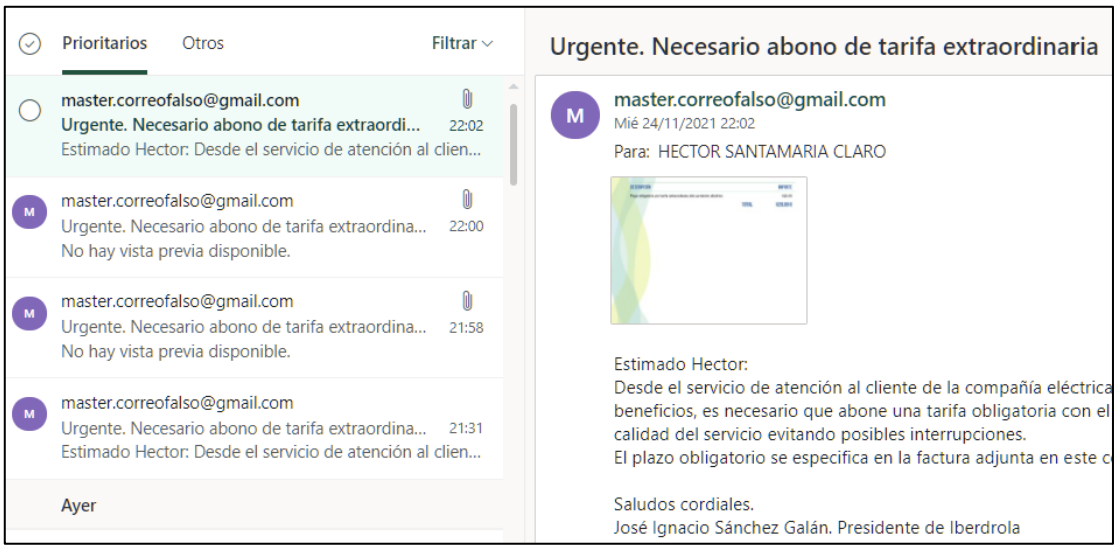


Figura 3.12 Mensaje Spear-Phishing en la bandeja de entrada.

Como se ha comentado en los párrafos anteriores, con un servidor y un dominio es posible utilizar las herramientas Postfix para crear un servidor SMTP o Dovecot para un servidor IMAP y recibir el correo, por ejemplo, en el cliente Outlook en una máquina dentro de la misma red. Al configurar Postfix para su funcionamiento en una máquina virtual dentro de la red junto con algunos parámetros como el dominio y el host, Outlook funcionaría con la configuración de cuentas personalizada. Algunos detalles de su configuración en el servidor Ubuntu del entorno virtual explicado en el apartado Metodología del trabajo, se muestran en la figura 3.13.

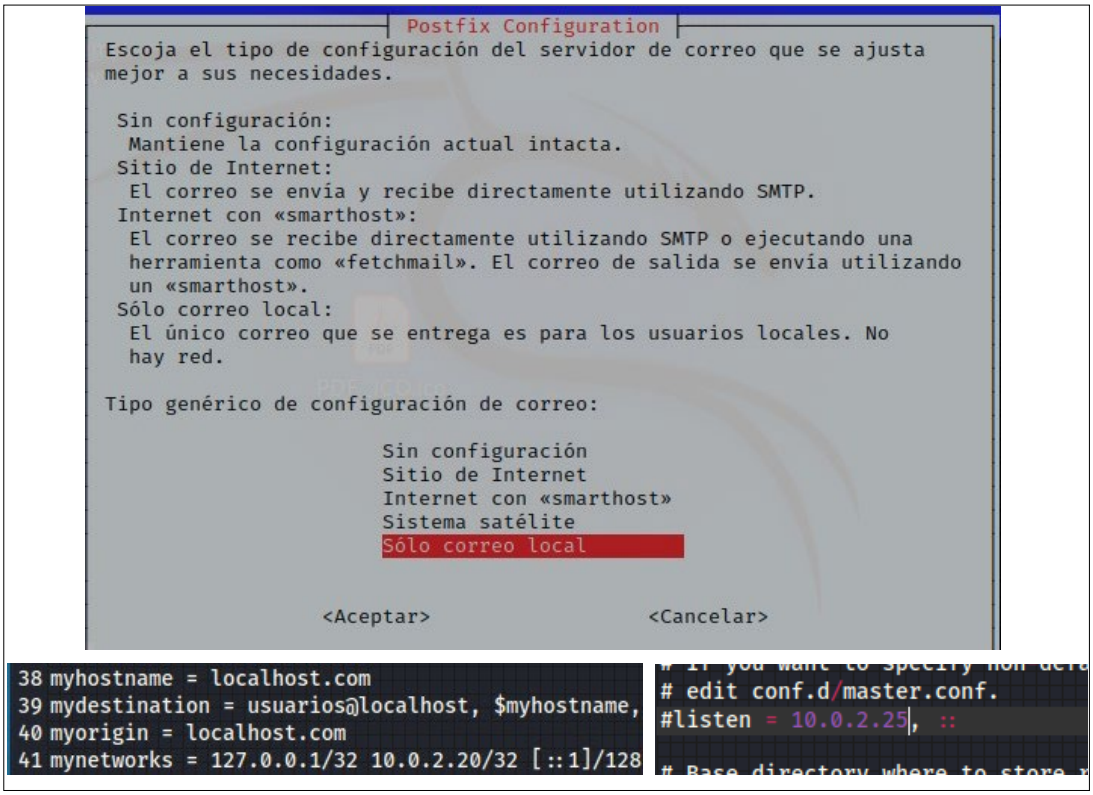


Figura 3.13 Capturas de pantalla del proceso de instalación y configuración de Postfix y Dovecot.

Con este procedimiento y después de haber modificado el perfil de remitente de Gophish para adaptarlo a la configuración del servidor SMTP creado con Postfix, el mensaje sí puede llegar a un cliente de correo como Outlook. La modificación de las cabeceras del mensaje también es válida porque ha cambiado a *clientes@iberdrola.com* y el fichero adjunto, que en realidad se trata del ejecutable autoextraíble modificado con cambios Unicode (RTLO) en el nombre, aparece sin ninguna restricción en la figura 3.14.

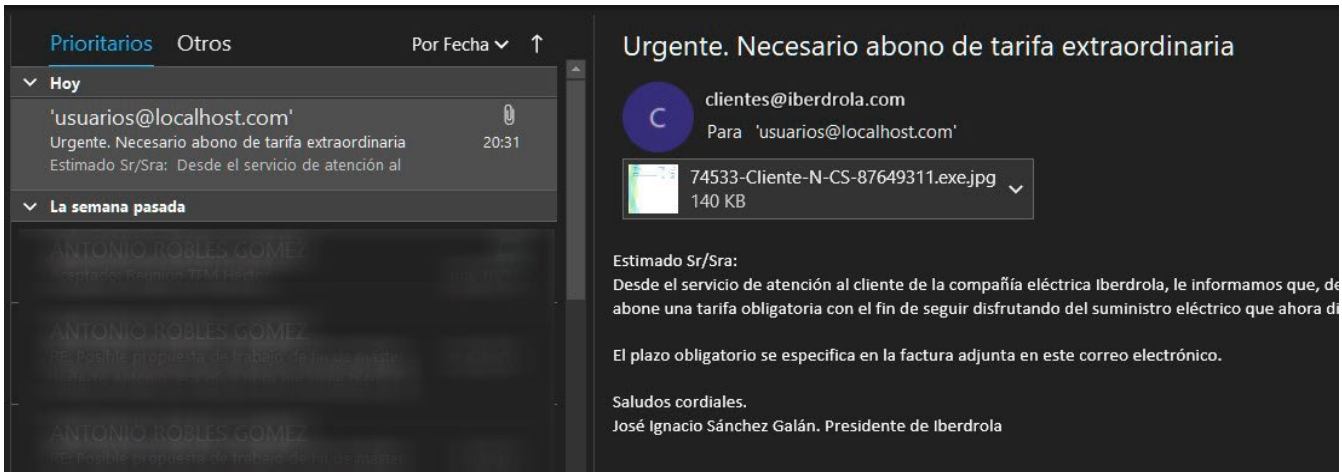


Figura 3.14 Correo electrónico con malware de campaña Spear-Phishing enviado con éxito.

4. Análisis estático

Para realizar el análisis estático se parte de la muestra de ransomware obtenida perteneciente a la familia Ryuk. En esta fase del análisis de malware se recaba información sobre el malware sin proceder a su ejecución, extrayendo características inherentes al tipo de archivo, en este caso de tipo PE para Windows. Los datos recabados pueden aportar un conocimiento significativo, sirviendo, además, para facilitar la comprensión de lo que se descubra en las siguientes fases: Análisis dinámico y Análisis de código.

4.1- Aspectos generales: Hashes y datos de compilación

Aun conociendo la procedencia de la muestra analizada, es conveniente recopilar los distintos tipos de hashes. Un escaneo del espécimen en VirusTotal devuelve las correspondientes cadenas.

Nombre de archivo
40b865d1c3ab1b8544bcf57c88edd30679870d40b27d62feb237a19f0c5f9cd1.exe

Tamaño de archivo
201,136 bytes

Registrado por primera vez
2022-01-28 10:28:24 UTC

MD5
484a2bcb1335ac97ee91194f4c0964bc

SHA-1
ad11ed52ab33ad05eb9b1e9ade134ca1348acc81

SHA-256
40b865d1c3ab1b8544bcf57c88edd30679870d40b27d62feb237a19f0c5f9cd1

Vhash
025046655d151098z5bhz2bza7z

Authentihash
016d028d571776f68fc0a8fe1d1c342da0553c0cc21c42c13273b434514598cd

Imphash
b8618e50f04dad9a118a51e1abfe6e25

Rich PE header hash
cad487fb7f20a1ad0ce898804f391a1d

SSDEEP
3072:08CBJvnmQ4VZQY83XS/cIVVEn+GNi4qRGE95jq:RWJOzT+K5Vc+oujq

TLSH
T1C4147B0175C18472E072293241B7CB725B3F7C303A61999B63B422BA5EB55D4AE38F3E

El hash difuso, denominado *SSDEEP* en VirusTotal, resulta de utilidad a la hora de analizar posibles familias y variantes de este ransomware, ya que permite establecer comparaciones de similitud generando un porcentaje. Cada una de las cuatro secciones, incluida la cabecera, está identificada con estos hashes escritos en la tabla 4.1. Es destacable que el archivo PE contiene una sección *Overlay*. Su contenido suelen ser datos que se agregan al final del archivo.

Nombre	Offset	Tamaño	Hash
PE Header	00000000	00000400	ad8646e8bd7dc47d77c6b5f863181c95
Section(0)[.'text']	00000400	00016e00	88921c85e4f2615829d9008f5fedebe8
Section(1)[.'rdata']	00017200	00007800	4b65a4e9bc64590a26e388f8f1c2ff74
Section(2)[.'data']	0001ea00	00011800	c276e592f933b546ee7134ab9c00372c
Section(3)[.'gfids']	00030400	00000200	5196364cf86bdf422bfe283199d72153
Overlay	00030600	00000bb0	610a761200f1857b795e246c070f3a3f

Tabla 4.1 Hashes de cada elemento en la tabla de secciones.

Independientemente de cómo se intente camuflar el nombre del archivo, un simple comando en Linux o un vistazo al código hexadecimal revelan, como muestra la figura 4.1, que se trata de un archivo ejecutable para Windows al coincidir la firma de la cabecera del archivo con los valores estándar.

```
~/Desktop$ file Ryuk_muestra

Ryuk_muestra: PE32 executable (GUI) Intel 80386, for MS Windows
```

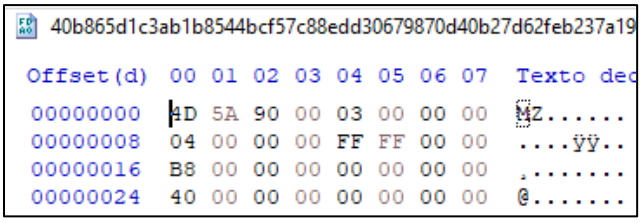


Figura 4.1 Evidencia de que se trata de un fichero PE para Windows.

El ransomware está compilado para una arquitectura de 32bits y esto se debe tener en cuenta, ya que usará el subsistema de Microsoft Windows-32-on-Windows-64 (WOW64) que posibilita la ejecución de aplicaciones de 32bits en sistemas operativos de 64bits.

Por otra parte, la información de cabecera del fichero compilado especifica en el parámetro *MajorOperatingSystemVersion* la versión correspondiente a Windows XP, aunque el archivo es mucho más actual. La herramienta Detect It Easy informa sobre las características del binario y el tipo de compilador:

```
filetype: PE32
arch: I386
mode: 32
endianess: LE
type: GUI
  compiler: EP:Microsoft Visual C/C++(2013-2017)[EXE32]
  compiler: Microsoft Visual C/C++(2015 v.14.0)[-]
  linker: Microsoft Linker(14.0, Visual Studio 2015 14.0*)[GUI32]
```

Complementando la información con la herramienta PEStudio y CFFExplorer, además de tener una arquitectura de 32 bits, está escrito en C++ con Microsoft Visual Studio 2015 en su versión 14.0.3. Otras características relevantes que aparecen en la figura 4.2, indican que aprovecha el subsistema de Windows de interfaz gráfica de usuario GUI (IMAGE_SUBSYSTEM_WINDOWS_GUI - 2) y, también en la cabecera opcional, el parámetro DllCharacteristics tiene añadidas las constantes:

- **NX_COMPAT**: Significa que la imagen PE es compatible con la tecnología NX (No eXecute). Este valor influye en la política DEP de Windows, de tal forma que, si la política DEP está establecida en OptIn, los procesos sin el indicador NX_COMPAT no obtendrán la prevención de ejecución de datos. Sería preocupante si dicho valor estuviese desactivado, ya que permitiría que Ryuk ejecutara código desde áreas de memoria prohibidas, por lo que esas funciones pueden no ser el objetivo del ransomware aquí analizado.

- **TERMINAL_SERVER_AWARE**: Al estar activada la opción evita que Terminal Server realice cambios en la aplicación para su correcto funcionamiento en entornos multiusuario. Estos

cambios implican modificaciones en el registro de Windows y la creación de ciertas carpetas virtuales.

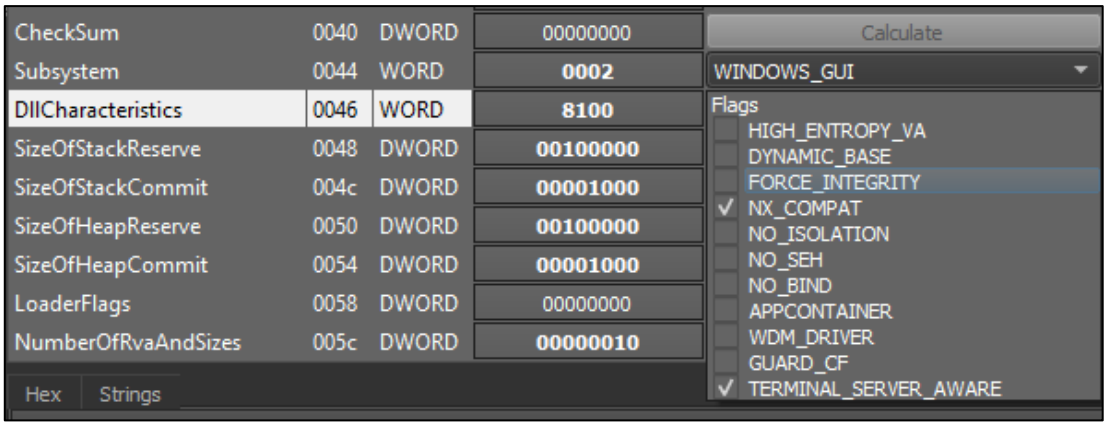


Figura 4.2 Constantes incluidas en la sección de la cabecera opcional "DllCharacteristics" mediante Detect It Easy.

Terminando con en estas observaciones iniciales, aunque el dato puede ser engañoso debido a la facilidad con la que puede ser falseado, el año de compilación data de enero de 2020. La figura 4.3 captura la información general que pestudio recopila de la muestra de Ryuk junto con la fecha de compilación.

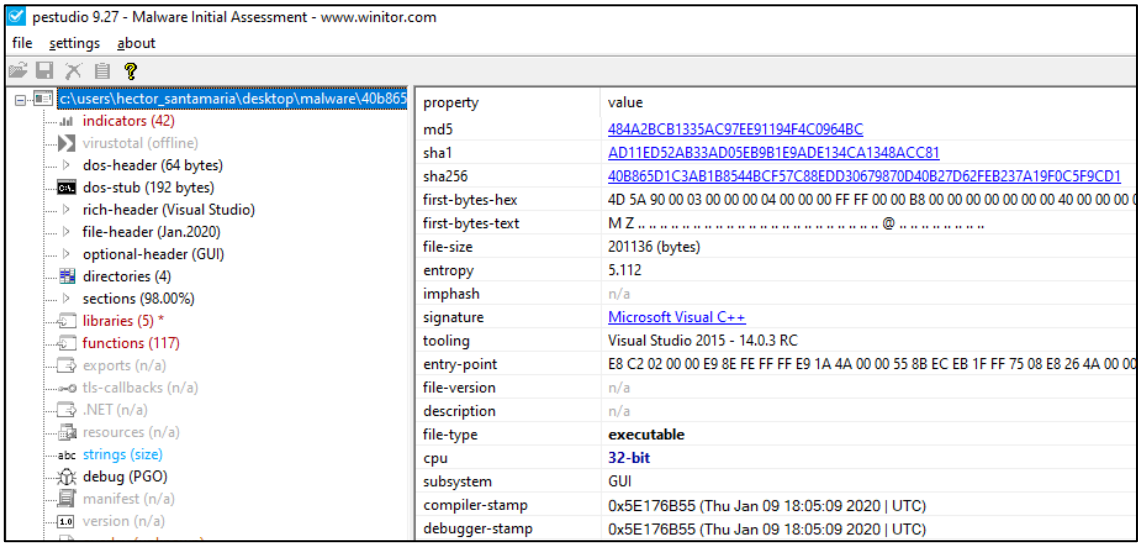


Figura 4.3 Información general de la muestra Ryuk que muestra el lenguaje, la versión del compilador y las fechas de compilación.

4.2- Empaquetado y secciones

El archivo original no contiene ningún empaquetado o cifrado y los análisis con la herramienta PEstudio y Detect It Easy no revelaron que poseyera ningún fichero de recursos porque la sección PE *resources* se encuentra vacía. Estos recursos pueden ser documentos PDF, Word o TXT que se usarían para dejar un mensaje a la víctima o camuflar la ejecución en segundo plano abriendo un archivo de texto cualquiera.

Existen, sin embargo, métodos manuales para que el analista pueda deducir si se ha empleado un *packer* o *crypter*. Uno de ellos es examinar el tamaño en bytes de las secciones en bruto y compararlas con el tamaño virtual. Si ambas tienen un valor cercano significa que no hay compresión ninguna, pero si el tamaño el bruto es mucho menor que el tamaño virtual, indica que se ejecuta alguna rutina de desempaquetado depositando en memoria los datos descomprimidos con su tamaño real.

property	value	value	value	value
name	.text	.rdata	.data	.gfids
md5	77F776DBD33435C545AC5C...	516AB2CFFAF280BACE1CC2...	5F603030DC91CF95A480471...	9D4AE0E8DBD73AD3826C28...
entropy	6.621	5.290	1.843	1.739
file-ratio (98.00%)	46.58 %	15.27 %	35.89 %	0.25 %
raw-address	0x00000400	0x00017200	0x0001EA00	0x00030400
raw-size (197120 bytes)	0x00016E00 (93696 bytes)	0x00007800 (30720 bytes)	0x00011A00 (72192 bytes)	0x00000200 (512 bytes)
virtual-address	0x30001000	0x30018000	0x30020000	0x3016F000
virtual-size (1494171 bytes)	0x00016C27 (93223 bytes)	0x00007630 (30256 bytes)	0x0014E968 (1370472 bytes)	0x000000DC (220 bytes)

Figura 4.4 Secciones mostradas por PEStudio junto con su tamaño en bruto y virtual.

En este caso, en la figura 4.4 se aprecia que la sección *.data* presenta una diferencia bastante acusada entre los dos tamaños, el real y el asignado en la memoria virtual. Como en parte sugieren Kleymenov y Thabet (2019) y las observaciones del propio alumno, la anomalía tiene explicación ya que en *.rdata* se almacenan cadenas de texto, constantes e información sobre las importaciones siempre como solo lectura, mientras que *.data*, al no disponer la muestra de sección *.bss*, puede contener variables con punteros a otros valores que no se asignarán hasta que se ejecute el programa.

Respecto a la entropía, cuanto más cerca de 8.0 se encuentre su valor, más probable es que exista algún tipo de compresión. (Mohanta y Saldanha, 2020). La entropía en el

ransomware analizado es significativamente baja, con un valor correspondiente a 5.11, sin embargo, presenta unos valores relativamente altos en la sección *.text*, que contiene el código del programa y esto hace que Detect It Easy la califique como empaquetada. Este hecho se detalla en la figura 4.5.

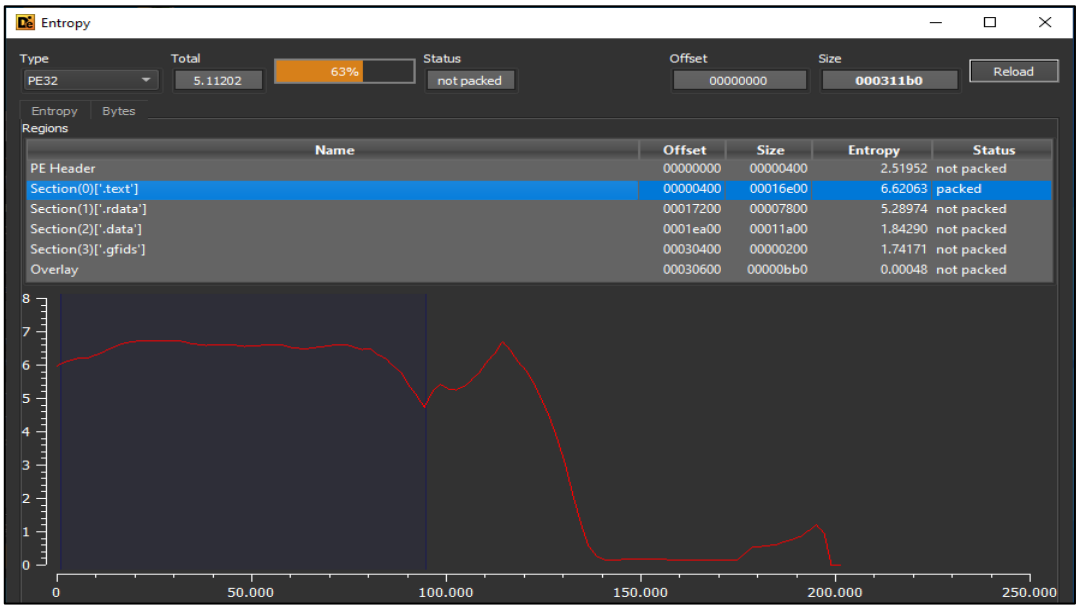


Figura 4.5 Valores de entropía divididos por secciones.

Para cada una de las secciones existe un campo *characteristics* con una composición en hexadecimal de *flags* cuyos valores indican las características de cada sección. La tabla 4.2 explica el significado de los que se han encontrado. En este caso, corresponden con cualquier archivo típico Portable-Ejecutable.

Sección	Valor decimal	Valor hexadecimal	Significado flags
.rdata y .gfids	1073741888	40000040	0x40000000 IMAGE_SCN_MEM_READ La sección puede ser leída. _____ 0x00000040 IMAGE_SCN_CNT_INITIALIZED_DATA La sección contiene datos inicializados

.text	1610612768	60000020	0x20000000 IMAGE_SCN_MEM_EXECUTE La sección se puede ejecutar como código. + 0x40000000 IMAGE_SCN_MEM_READ La sección puede ser leída. _____ 0x00000020 IMAGE_SCN_CNT_CODE La sección contiene código ejecutable
.data	-1073741760	C0000040	0x40000000 IMAGE_SCN_MEM_READ La sección puede ser leída. + 0x80000000 IMAGE_SCN_MEM_WRITE Se puede escribir en la sección. _____ 0x00000040 IMAGE_SCN_CNT_INITIALIZED_DATA La sección contiene datos inicializados

Tabla 4.2 Significados de flags hallados en las características de cada sección.

No se encontraron aspectos más relevantes en las secciones y, al menos en el análisis estático, nada indica que la muestra examinada actúe como un *packer*:

- Las dependencias a librerías y funciones de la tabla *imports* se muestran con normalidad.

- La diferencia entre el tamaño en disco y el tamaño virtual es correcta, salvo en el comentado caso de la sección *.data*.

- La entropía se encuentra dentro de un rango aceptable.

4.3- Dependencias a librerías

Ryuk aprovecha los recursos del sistema operativo mediante llamadas a librerías por enlace dinámico. La sección *imports* que muestra herramientas como PE-Bear o PEstudio como se observa en la figura 4.6, indica la existencia de estas dependencias:

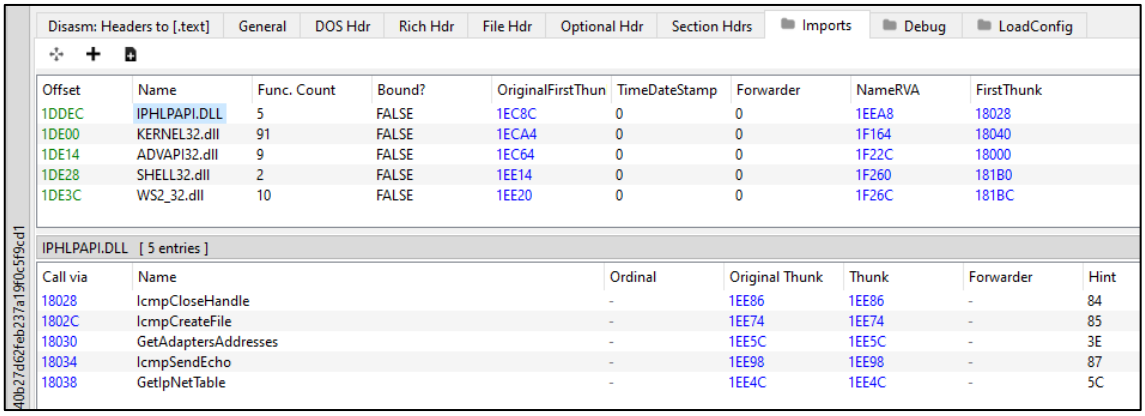


Figura 4.6 Librerías DLL y sus correspondientes llamadas a funciones mostradas por PEBear.

4.3.1- iphlpapi.dll (IP helper application programming interface)

La inclusión de esta DLL implica que el ransomware posee funciones que le permiten trabajar con Windows IP Helper API y por lo tanto con protocolos TCP/IP (Transmission Control Protocol/Internet Protocol) , enrutamiento o SNMP (Simple Network Management Protocol) para interactuar con dispositivos en la red. Las funciones contabilizadas de las que hace acopio el ransomware sobre iphlpapi.dll son bastante reveladoras en cuanto a sus capacidades de propagación:

- **IcmpCloseHandle, IcmpCreateFile e IcmpSendEcho**: crean un manejador o handle para que puedan emitirse solicitudes “Echo” mediante el protocolo ICMP (Internet Control Message Protocol) bajo IPv4. Esto se puede utilizar para detectar otras computadoras en la red local.

- **GetAdaptersAddresses**: Devuelve las direcciones IP asociadas a cada adaptador de red. Esto se realiza a nivel local, luego es de interés para Ryuk conocer esa información. Además, esta función también recupera otra información como la interfaz de la máquina (direcciones *unicast*), direcciones IP de los servidores DNS (Domain Name System) o servidores WINS (Windows Internet Name Service).

Como se verá más adelante, este ransomware está diseñado para propagarse por la red e implementa las funciones necesarias para el descubrimiento de otras redes locales.

4.3.2- Kernel32.dll

Proporciona funcionalidades elementales para interactuar con el sistema operativo, desde ficheros y elementos del sistema de archivos al control sobre los procesos y acceso a memoria. La muestra seleccionada de Ryuk invoca a 91 funciones del kernel de Windows entre las que, a parte de las típicas que dan soporte a la creación, copiado y borrado de archivos, destacan las siguientes:

- **GetLogicalDrives**: Realiza un mapeo de las unidades de almacenamiento lógicas presentes en el sistema operativo. El objetivo del ransomware es cifrar el contenido del dispositivo que los autores hayan programado en su código, por lo que es de utilidad ese movimiento por todas las unidades presentes. Según el valor retornado de *GetLogicalDrives*() se determina si la "Unidad X:" Existe y es accesible, o no.

- **TerminateProcess, CreateRemoteThread y otras de processthreadsapi.h:** Existen varias funciones relacionadas con el manejo de procesos. Son notables *TerminateProcess*, que le permite finalizar un proceso cualquiera siempre que posea permisos de administrador y *CreateRemoteThread* que facilita crear un proceso dentro del espacio de direcciones virtual de otro proceso cualquiera. Esto sugiere dos cosas, que Ryuk intenta finalizar procesos de aplicaciones que no le interesan y que cuenta con capacidades de inyección.

  Sin embargo, llama la atención el hecho de que no disponga de llamadas a *EnumProcesses*

o *EnumProcessModules* que devuelven información sobre los procesos en ejecución y los módulos cargados en ellos. Lo que revelan estas llamadas a funciones es que el ransomware no hace intentos por poseer una información detallada en ese aspecto, sino que crea de procesos o finaliza otros.

- **WriteconsoleW GetConsoleCP, GetConsoleMode**: Son tres funciones que se utilizan para acceder a una consola e interactuar sobre ella. En concreto las tres mencionadas realizan funciones relacionadas con el manejo del búfer y la escritura en una consola. Existe la posibilidad de que esté haciendo uso del componente WMI (Windows Management Instrumentation) para lanzar consultas WQL y llevar a cabo sus acciones maliciosas, pero al haber incluido los autores *GetCommandLineA* entre las funciones invocadas también es muy probable (y será confirmado en el apartado de Dependencias a librerías e Interacciones con el sistema operativo) que se trate de la consola de comandos de Windows o CMD

4.3.3- Advapi32.dll (Advanced Windows 32 Base API)

La librería advapi implementa utilidades avanzadas relacionadas con el registro de Windows, servicios y usuarios. Las funciones que el ransomware integra arrojan datos relevantes sobre su manera de proceder en cuanto a privilegios y persistencia principalmente.

- **OpenProcessToken, OpenThreadToken, GetTokenInformation:** Las funciones permiten obtener de un proceso un token de acceso. Ayudan al malware a realizar el trabajo de gestión de procesos e hilos comentado anteriormente. Esta operación requiere de ciertos privilegios para el programa o proceso que lo solicita. *GetTokenInformation* posibilita el trabajo con el token al suministrar información sobre este.

- **AdjustTokenPrivileges, LookupPrivilegeValueW**: Relacionado con las funciones comentadas anteriormente, es mediante esta función como se habilitan o deshabilitan los privilegios de acceso que se tienen sobre el token. *LookupPrivilegeValueW* le ayuda a recuperar el identificador asociado al nombre del privilegio que se está manejando.

Si Ryuk opera como un usuario elevado, podría con *AdjustTokenPrivileges* activar o desactivar privilegios para finalizar el proceso asociado al token.

- **LookupAccountSidW**: Función importante en lo que se refiere al descubrimiento de cuentas de usuario o de dominio disponibles en el sistema. De una lista de SIDs (Identificadores de seguridad que identifican a usuarios o a grupos) recibe nombres de cuentas en la red y locales. Es una forma de proceder para la propagación dentro del mismo sistema o en la red de área local.

- **OpenSCManagerW, EnumServicesStatusW**: Como recoge la documentación oficial para Desarrolladores de Apps Windows, Microsoft (2021), *OpenSCManager* establece una conexión con el administrador de control de servicios en la computadora especificada y abre la base de datos del administrador de control de servicios especificado, lo cual permitiría a *EnumServicesStatusW* recopilar información sobre los servicios disponibles, además de información como su estado (inactivo, activo), nombre o si pertenece al Kernel o al espacio de aplicaciones.

Se debe resaltar que el binario incluye funciones como *GetCurrentThread*, *OpenProcessToken*, *OpenThreadToken*, *AdjustTokenPrivileges*, *LookupPrivilegeValue* junto con *ImpersonateSelf*, que no se han explicado, cuya combinación está típicamente asociada a la escalada de privilegios y al ajuste de los tokens de seguridad para que un proceso actúe con permisos de administrador. (Mohanda y Saldanha, 2020).

4.3.4- shell32.dll

Shell32 proporciona múltiples utilidades a Windows, pero examinando las dos únicas funciones que los autores del ransomware aprovecharon, se deduce que puede crear comandos y abrir ficheros ejecutables además de examinar el contenido de cualquier carpeta sobre la que tenga acceso :

- **ShellExecuteW**: Realiza operaciones con el archivo o carpeta que recibe como parámetro, concretamente un archivo ejecutable y esto le confiere capacidades para hacer uso de cualquier aplicación del sistema, aunque también, si así lo especifica el programa, puede explorar o buscar contenido en un directorio.

- **CommandLineToArgvW**: Cuando se escribe una línea de comandos, CommandLineToArgv la formatea para convertirla en un comando válido que pueda ser interpretado por el sistema operativo.

Un ejemplo sería escribir:

`Programa.exe -crypt c:/miProyecto.docx` que creará tres punteros a estas tres cadenas de texto "Programa.exe", "-crypt", "c:/miProyecto.docx"

Los punteros se podrían enviar a ShellExecuteW indicando los punteros a string cuyo valor en la instrucción se representaría así:

```
ShellExecute(NULL, "open", Programa.exe, [-crypt][c:/miProyecto.docx], NULL, SW_HIDE);
```

4.3.4.1- ws2_32.dll (Windows Sockets 2)

Winsock es una librería de enlace dinámico que permite trabajar con las redes de internet, intranet u otras aplicaciones que transmitan información de forma cableada independientemente del protocolo. Está considerada una librería "sospechosa", ya que muchos malware la aplican para llevar a cabo comunicaciones con servidores u otros dispositivos en la red. Algunas herramientas de análisis estático como PEBears y Detect It Easy no son capaces de identificar las funciones ya que los desarrolladores han tratado de ocultarlas. PEStudio, como se aprecia en la figura 4.7, sí muestra los nombres y son los siguientes:

- **Socket, closesocket, setsockopt, bind:** Para crear conexiones TCP, ICMP, UDP (User Data Protocol) entre otros, Ryuk emplea estas funciones**.** *Socket*, valga la redundancia, abre un socket según el protocolo especificado y *closesocket* lo cierra. Por otro lado, *setsocketopt*

posibilita configurar un socket existente con parámetros tales como habilitarlo para enviar paquetes *Keep-alive*, el tamaño del búfer o el número de segundos que esperará una conexión TCP una respuesta por parte del otro extremo. La dirección local objetivo se asocia al socket con *bind.*

- **WSAStartup, Sendto, WSACleanup** : Primero, con *WSAStartup* se inicializa el uso de *winsock* (ws2_32.dll) por parte de un proceso, después *Sendto* envía datos a un destino específico, que requiere el identificador del socket creado anteriormente y en el último paso, WSACleanup da por terminado el uso de Winsock en el programa.

- **inet_addr**: Su trabajo es convertir una cadena de texto con una dirección IP con puntos similar a esta 192.168.1.1, en una notación adecuada como unsigned long.

- **Htonl, htons**: Transforma el número de 32 bits que compone una dirección IPv4 en formato u_long (unsigned long) en el caso de htonl y u_short en el caso de htons, al formato que se usa en las redes TCP/IP. Esto indica que Ryuk utiliza el protocolo TCP a nivel de capa de red para las comunicaciones.

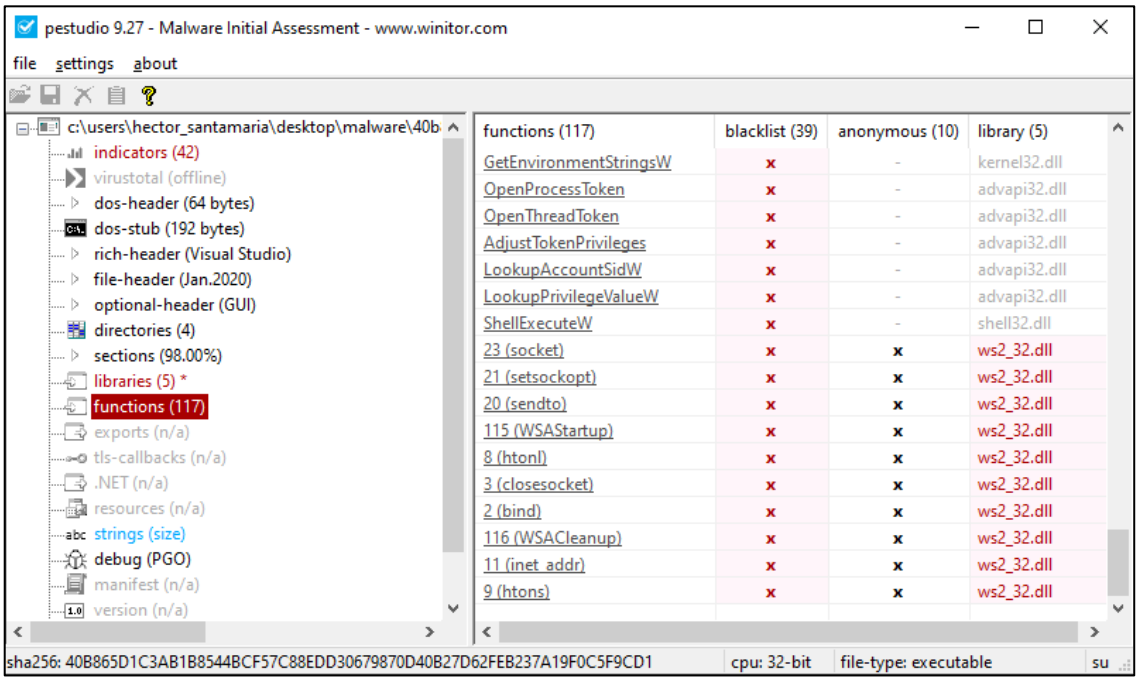


Figura 4.7 Llamadas a funciones de la librería ws2_32.

Aunque este tipo de funcionalidades podrían estar presentes en cualquier otra aplicación con conexión a internet desarrollada para Windows, en el malware representa con total certeza que crea conexiones o bien hacia un servidor C2 o bien hacia otros dispositivos a su alcance por medio de la red. En el caso del ransomware, esto último es lo más probable.

Otra observación quizá no tan relevante es el uso de *htonl* y *htons*. Entendiendo que al manejar una dirección IP lo lógico es obtener un unsigned long devuelto por la función *inet_addr*, este malware debe estar usando rangos de direcciones concretos que requieren de la función htons para traducir, por ejemplo, una variable de tipo short (con un tamaño máximo de 16 bits) al formato admitido por TCP/IP. Si la intención fuera sondear rangos de direcciones privadas entre “10.0.0.0” a “10.0.255.255”, debería almacenarse “10.0” en memoria que luego pasaría a manos de la función *htons*.

4.4- Cadenas de texto (strings)

Dentro muchos ficheros ejecutables se hallan cadenas de texto que pueden proporcionar información muy relevante sobre el comportamiento del programa. Para los desarrolladores es necesario incluir texto si necesitan especificar un rango de direcciones IP, el nombre de un elemento, alguna URL o la ruta relativa o absoluta donde se van a almacenar ciertos artefactos. Herramientas como BinText o PEStudio-CLI son de utilidad para exportarlas. La figura 4.8 revela algunas de estas cadenas de texto o *strings* en BinText o el resultado de su exportación con PEStudio-CLI mostrado en la consola de comandos de Linux.

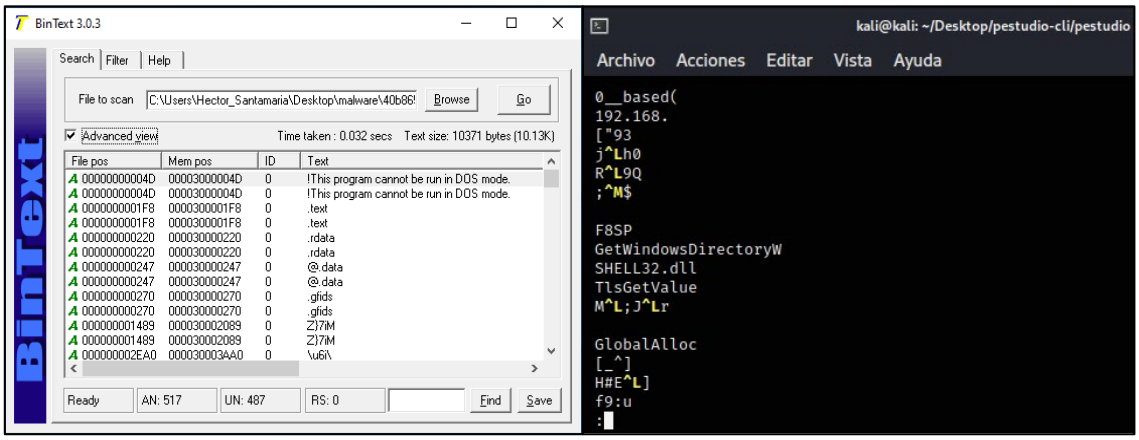


Figura 4.8 Herramientas que permiten visualizar y exportar las cadenas de texto de Ryuk.

En el caso de Ryuk la mayoría de estos *strings* no están cifrados y los más relevantes son los descritos a continuación clasificados en secciones.

4.4.1- Capacidades criptográficas

Como cualquier ransomware Ryuk incorporará funciones y métodos criptográficos para el cifrado de la información. No será hasta las siguientes fases de análisis cuando se conozca su forma de proceder al respecto, pero las cadenas de texto encontradas ya aportan cierto conocimiento.

- Microsoft Base Cryptographic Provider v1.0

  Según la Documentación para Desarrolladores de Microsoft (2021), es un proveedor de servicios criptográficos distribuido con las versiones 1.0 y 2.0 de CryptoAPI, que proporciona utilidades de autenticación, codificación y encriptación a aplicaciones Windows. Esta cadena hallada indica que hace uso de la API de Microsoft para trabajar con RSA.

- Encrypt y RSA1

  La palabra "encrypt" se puede obtener del listado de strings y representa el nombre de una función que provee la Api de Windows *CryptoApi*. No existe ninguna referencia a la acción contraria de desencriptado. Por otro lado, es posible encontrar el formato en el que se encuentra la clave RSA, que es compatible con el utilizado por Microsoft. En una de las cadenas de texto aparece RSA1, seguido de más caracteres distribuidos en direcciones de memoria contiguas que conforman la clave pública.

  En la figura 4.9, está descrito como, siguiendo la posición en el archivo para un ancho de columnas de 8 bytes (20208h) en un editor hexadecimal como HxD, la clave se observa más claramente.

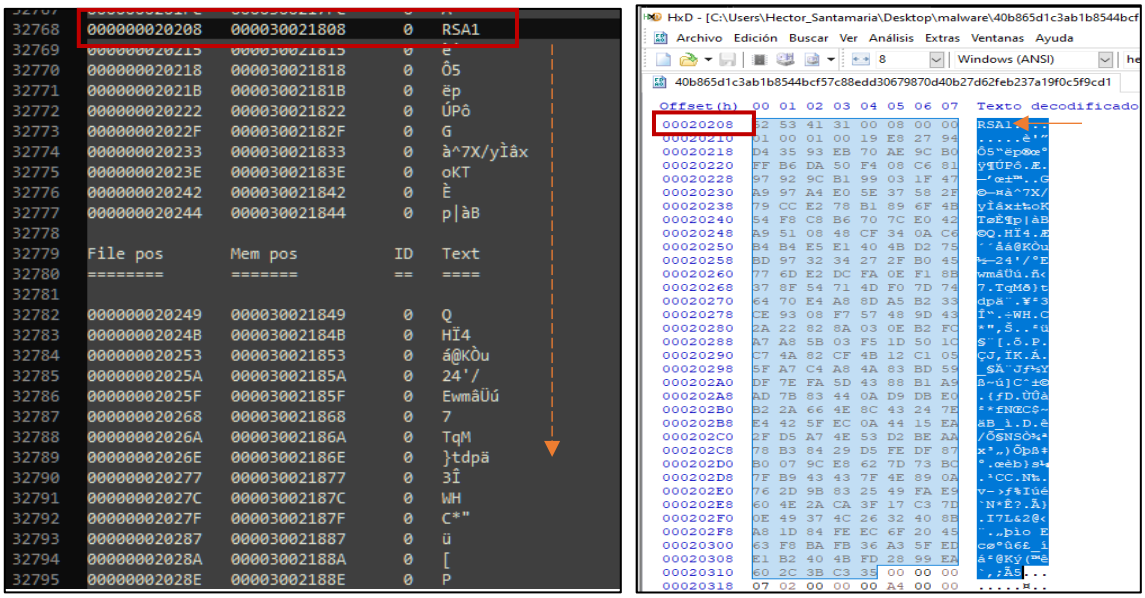


Figura 4.9 Clave pública detectada como "string" en el fichero binario y visualizada luego en editor hexadecimal.

El formato empleado permite diferenciar cada una de sus partes:

- **RSA1**: Es una clave pública, de lo contraría el valor sería *RSA2*
- **800h:** La longitud del módulo igual a 2048 bits
- **10001h**: El exponente público igual a 65537
- **19E82794D43593EB…h**: El módulo RSA de la clave pública

4.4.2- Sistema de archivos

Ryuk recorre el sistema de archivos de la máquina infectada para cifrar los ficheros. Es notable el hecho de que entre las cadenas de texto no se encuentran extensiones de archivos. Algunos ransomware como Wannacry tienen como objetivo ciertos tipos de archivos con casi siempre información relevante para el usuario u organización, como pueden ser documentos de office (*.docx, xlsx*) o imágenes (*.jpeg, .png, .tiff*).

- `"\*.*", "..", "\Windows\"`

  Incorpora los caracteres necesarios para moverse entre directorios o ejecutar un comando, como son *"/"*, *"\"*, *".."* para retroceder o "**.**" para acceder a una ruta específica y cifrar todos los archivos disponibles. El daño causado es importante, ya que impide el correcto funcionamiento de las aplicaciones instaladas.

  También incluye el nombre del directorio *Windows* ya que, como se detallará posteriormente, evita cifrar el contenido de la carpeta con el sistema operativo con el fin de permitir el acceso a la computadora y que esta sea mínimamente operativa.

- `\Documents and Settings\Default User\finish - \users\Public\finish`

  Rutas escritas pensando en versiones antiguas de Windows. En Windows XP la dirección se correspondería con *"C:\Documents and Settings\Default User\finish"* mientras que a partir de Windows Vista el equivalente es *"C:\Users\Public\finish"*. De cualquier modo, las versiones actuales de Windows accederán a los mismos directorios con cualquiera de los dos indistintamente. Los creadores han especificado dos carpetas "*\finish*" y "*\sys*" para las dos rutas.

- `.RYK`

  Estos caracteres conforman la única extensión claramente identificable detectada en la extracción de *strings*. Una vez finalizado el proceso de cifrado, a los archivos se les asignará esta extensión.

4.4.3- Interacción con el sistema operativo

Las cadenas texto revelan algunos elementos del sistema operativo con el que Ryuk interactúa.

- `\System32\cmd.exe`

  La ruta hacia la conocida consola de comandos de Microsoft se encuentra especificada en el código fuente. En las librerías y funciones importadas descritas en el apartado *Dependencias a librerías*, ya aportaban evidencias del procesado que debían seguir las cadenas de caracteres para su escritura y posterior ejecución en la consola de comandos.

- `icacls /grant Everyone:F /T /C /Q`

  El comando icalcs, que sustituye a *cacls* disponible en versiones anteriores a Windows Vista, permite examinar y modificar la llamada ACL. La ACL es la lista de control de acceso que determina los permisos asignados a los archivos y carpetas de Windows.

  Con la instrucción *"grant Everyone:F /T /C /Q"* Ryuk se asegura de que la unidad especificada en tiempo de ejecución y recopilada con la función *GetLogicalDrives*, establezca todo su contenido con permisos de acceso completo para cualquier usuario:

  - Puesto que */grant* no incorpora el modificador *":r"* → */grant[:r]*, los permisos se añadirán a cualquier otro otorgado anteriormente y no se reemplazarán por uno anterior.
  - *'F'* Concede acceso total a los directorios y archivos: Lectura, escritura, borrado y ejecución.
  - *"/T"* La operación se realizará para todos los archivos en todos los subdirectorios.

- *"/C"* Si hubiera errores al asignar los permisos en cualquier archivo, el proceso continuaría.
- El comando no incluye *"/L"*, por lo que el cambio de permisos en un enlace simbólico se llevará a cabo en el destino y no en el enlace.
- *"/Q"* Se suprime cualquier mensaje hacia el usuario cuando la operación haya finalizado con éxito.

- `cmd /c "WMIC.exe shadowcopy delet"` y `vssadmin.exe Delete Shadows /all /quiet`

  Los desarrolladores de este malware tuvieron en cuenta la característica *Shadow Copies* incorporada en Windows XP y gestionada por VSS (Volume Shadow Copy Service), que posibilita la creación de copias de seguridad de los archivos, así como los puntos de restauración. La figura 4.10 refleja los métodos encontrados en las cadenas de texto para tal propósito.

  Dichos métodos son muy habituales en la mayoría de las familias de ransomware y tienen como propósito, a parte de la recuperación de los archivos dañados, impedir revertir el sistema a un estado anterior. (Hunter. B, 2020).

  El procedimiento más utilizado es mediante *vssadmin* con el que se pueden gestionar todas las instantáneas creadas en un volumen:

  - El parámetro */all* eliminará todas las instantáneas de la unidad seleccionada.
  - El parámetro */quiet* ejecutará el comando de forma transparente al usuario, sin mensajes de ningún tipo.

  El segundo método es a través de *Windows Management Interface Command (WMIC)* utilizada para emitir comandos WMI (Windows Management Instrumentation), pero que también permite eliminar las Shadow Copy.

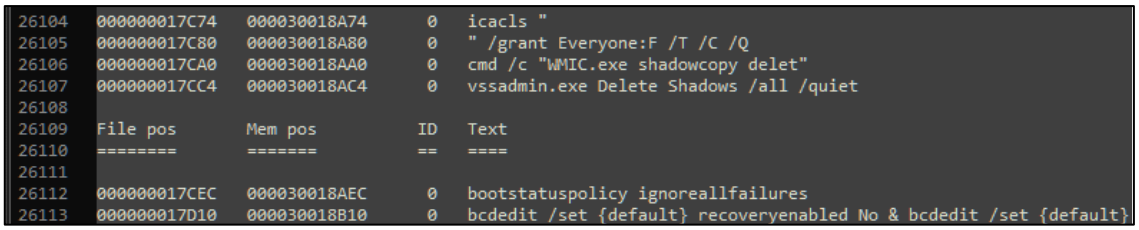


Figura 4.10 Detalle de los comandos utilizados para la eliminación de las Shadow Copies en posiciones contiguas de memoria y en archivo.

- `bcdedit /set {default} recoveryenabled No & bcdedit /set {default} y bootstatuspolicy ignoreallfailures`

  Las dos últimas cadenas de texto modifican el arranque de Windows alterando el BCD *(*Boot Configuration Data, base de datos con la configuración del proceso de arranque en Windows) a través de bcdedit, de forma que:

  - *Set recoveryenabled No*: establece el valor No a la recuperación automática de Windows durante el arranque, antes del inicio del sistema operativo. Con esta estrategia el usuario no podrá acceder al menú de recuperación que permitiría, entre otras cosas, restaurar el sistema a un estado anterior o iniciar el sistema en modo a prueba de fallos.
  - *bootstatuspolicy ignoreallfailures*: Configura al Sistema operativo para que ignore cualquier error de arranque e inicie Windows con normalidad.

  Siguiendo esta metodología, el ransomware evita por todos los medios que el sistema operativo ofrezca al usuario opciones de recuperación para revertir los efectos maliciosos o, incluso, evitar que paren al no poder reiniciar Windows en modo a prueba de fallos y tener acceso a una consola de comandos antes del arranque.

4.4.4- Movimiento lateral y funciones de red

Se detectan en la muestra analizada algunos indicadores sobre las funciones de red que implementa correspondientes al manejo de direcciones IP.

- `10. - 172.16. - 192.168.`

  Estos tres bloques de direcciones representan los rangos de IP reservados para las redes privadas. Concuerda con el uso de *htons* comentado en el apartado sobre la librería Winsock2 que provee capacidad para transformar los formatos desde un *unsigned short (u_short)* al que utiliza el protocolo TCP/IP. Ryuk, por lo tanto, está preparado para moverse entre todas las direcciones de red locales que detecte.

4.4.5- Evasión de software antivirus y procesos

Las cadenas de texto extraídas revelan multitud de nombres de procesos que son recopilados y mostrados en la tabla 4.3:

Nombre	Descripción
Software Antivirus	
Antivirus	Los ciberdelincuentes tuvieron en consideración incluir la palabra antivirus para de forma genérica detectar procesos que la incluyan.
Sofos	Empresa que desarrolla software y hardware de seguridad. La cadena de texto se encuentra mal indicada. El nombre de la compañía es Sophos y el proceso asociado a su antivirus también lo es. Podría tratarse de un error en un programador despistado.
Ekrn,	Proceso perteneciente a ESET Nod32 y a sus familias de antivirus.
Tmlisten, PccNTMon, CNTAoSMgr, Ntrtscan	Corresponden a los servicios y procesos del antivirus de Trend Micro
mbamtray	Archivo ejecutable y proceso del mismo nombre de MalwareBytes Anti-Malware
dcagent	Implicado en el funcionamiento de los antivirus y firewalls de aplicaciones de Fortinet.
Enpoint, EPSecurity, EPUpdate	Aplicación de seguridad de la familia *Endpoint Security* de *Bitdefender*. EPUpdate se trata del servicio de actualización del software de la compañía.
EsgShKernel	Asociado al Programa antispyware y antimalware *SpyHunter*
smc	Firewalls y Antivirus de Symantec
Procesos de Windows y de terceros	

Virtual, vmcomp y vmwp	Procesos implicados en el *Virtual Machine Management Service* cuando se llevan a cabo virtualizaciones con Hyper-V.
Veeam, ackup/backup/Backup	Nombre del proceso y software propietario *Veeam Backup* de copias de seguridad y recuperación para entornos físicos, virtuales y cloud. Utilizado como protección contra el ransomware. Una de las cadenas está incorrectamente escrita.
acronis	Software de *Acronis True Image* para soluciones de copias de seguridad.
bedbg	Al igual que otros nombres de procesos hallados, bedbg está relacionado con software para copias de seguridad, esta vez de *Symantec Corporation*
zoolz	Aplicación del servicio de copias de seguridad en la nube *Zoolz Cloud Backup*.
dbeng	Programa *BrightStor ARCserve Backup* de la compañía *ARCserve* cuyo propósito es garantizar la integridad de los datos en sistemas de almacenamiento de todo tipo a través de copias de seguridad.
calc	Corresponde a la calculadora de Windows
xchange	Perteneciente al editor de *PDF PDF-Xchange*.
excel, Outlook, onenote, powerpnt, word, visio, msaccess, infopath, mspub.	La mayor parte de los programas de la suite de Microsoft Office están referenciados en Ryuk
Sql, Oracle	Cualquier proceso relacionado con sistemas gestores de bases de datos podría verse implicado aquí. Por otro lado, incluye de forma explícita al proceso de Oracle.
Ocautoupds, ocomm	Procesos creados por Oracle que conectan el cliente de mensajería Outlook con bases de datos Oracle, servicios de almacenamiento u otros propios de Microsoft Windows
xfssvccon	Creado por el cliente de Oracle Drive, un programa antiguo utilizado en Windows XP o anteriores que sincronizaba ficheros almacenados en servidores o unidades remotas con el sistema operativo.
sqbcoreservice	Correspondiente a *SQL Backup Data Base* de los desarrolladores *Redgate*. Implica el funcionamiento de su programa de copias de seguridad de bases de datos para SQL Server.
encsvc	Servicio de *Citrix MetaFrame XP* entre otros. Posibilita conexiones hacia servidores Citrix. Pertenece a una versión antigua utilizada en Windows 2000 y XP.
steam	Proceso ligado a la popular plataforma de videojuegos *Steam*.
Thunderbird, tbirdconfig, thebat	Los primeros corresponden al cliente de correo gratuito de Mozilla y el último a otro cliente que garantiza funciones de privacidad y seguridad.

Tabla 4.3 Diferentes procesos externos y del sistema que tiene en cuenta Ryuk

Dentro de la categoría evasión de software antivirus, existe la referencia a una librería f54tgev4f3.dll relacionada con técnicas anti-depuración y anti-ingeniería inversa. (hybrid-analysis, 2019).

4.4.6- Persistencia

Las técnicas que este ransomware maneja para obtener persistencia y seguir operando aun después de un reinicio del sistema operativo pueden resumirse en un cambio en el registro de Windows.

- `/C REG ADD "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "svchos" /t REG_SZ /d /f"`

  Este string es parte de un comando que añade una entrada a la clave de registro *CurrentVersion\Run*. Las entradas *RUN*, están diseñadas para que los programas puedan autoejecutarse en cada inicio del sistema. Existen otras secciones dentro del registro de Windows donde pueden ubicarse, pero esta la forma en la que Ryuk lo hace.

  - /V: Especifica el nombre de la nueva entrada. El elegido por los desarrolladores es *svchos*, sin la "*t*" final, por lo que intentan que se asemeje al que crea Windows cada vez que pone en marcha un nuevo servicio: *svchost*.
  - /t: Indica cuál va a ser el tipo de entrada de registro. REG_SZ implica que será una cadena de texto Unicode o ANSI.
  - A continuación de */d* se debe añadir la ruta del fichero PE que el sistema operativo deberá ejecutar en el arranque. En esta fase del análisis no es conocida.
  - Concluye con el parámetro /f para añadir la entrada al registro sin un cuadro de diálogo de confirmación lo cual evita cualquier sospecha.

- `/C REG DELETE "HKEY CURRENT USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "svchos" /f`

  Del mismo modo que la muestra analizada añade una entrada *svchos*, también existe una cadena que la elimina.

- `/reg:64`

  En última instancia, reg:64 fuerza que la lectura y escritura se realice en la ubicación del registro de 64 bits.

  Puede verse en la figura 4.11 que en posiciones contiguas de memoria se localizan los caracteres "\\" y ".". La explicación procede de las funcionalidades del comando *REG ADD* entre las que se incorpora añadir o eliminar una clave de registro a un dispositivo remoto y para ello resulta imprescindible especificar la ruta de dicho dispositivo precedida de la doble barra invertida. Este hecho concuerda con las capacidades de red y movimiento lateral observadas en este apartado de análisis estático.

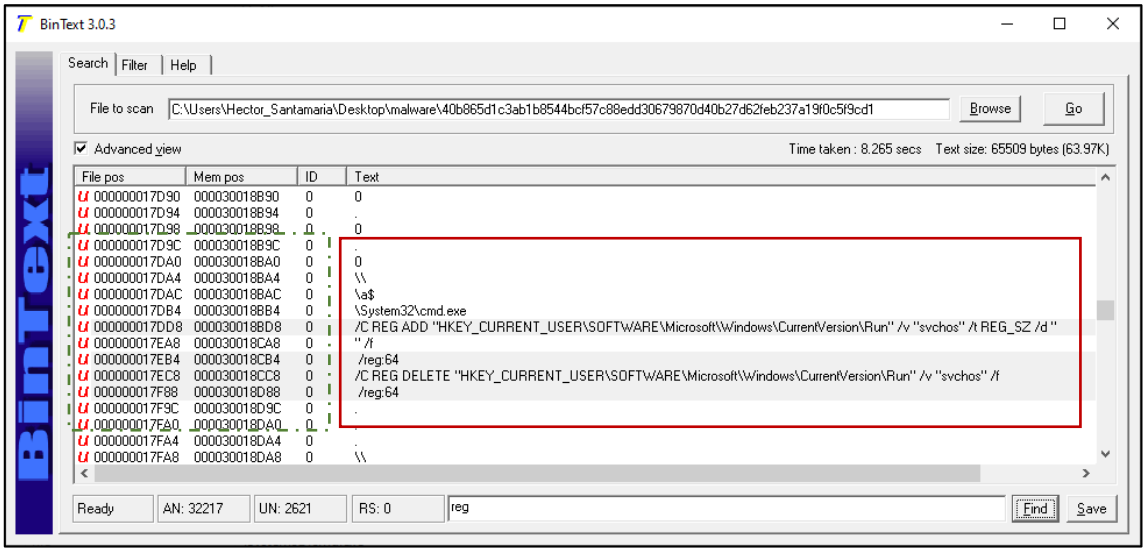


Figura 4.11 BinText mostrando los textos recuperados referentes al registro de Windows.

4.4.7- Otros

Otras cadenas de texto recuperadas igualmente llamativas son:

- `RyukReadMe.htm`

  Nombre de un archivo de hipertexto que, aunque con los datos estáticos no es posible estar al corriente de su ubicación final y el contenido, en posteriores fases se conocerán detalles y el propósito de su creación.

- `Texto cifrado de 5240 bytes`

  Dentro del archivo binario diseccionado en este trabajo, se encuentra embebida una larga cadena de texto de 5240 bytes (la más grande de todas las demás) cuyo contenido cifrado lo está para camuflar su detección por antivirus, reglas YARA o sistemas IDS/IPS basados en firmas.

  La figura 4.12 señala como en el código hay algunos otros strings cifrados también de cierta longitud y en formato ASCII.

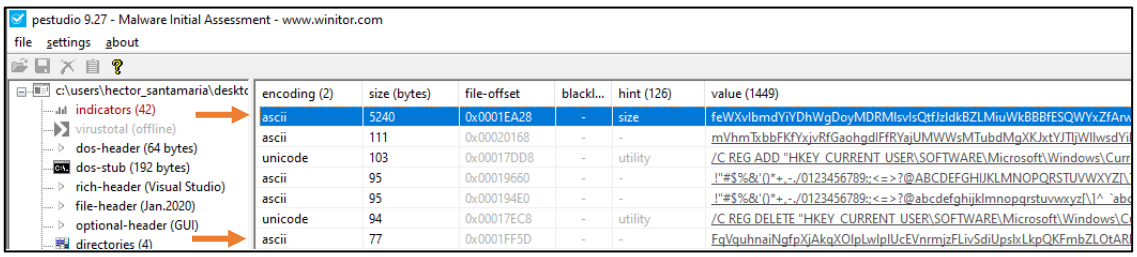


Figura 4.12 Cadena de texto cifrada de 5240 bytes.

- `SYSTEM\CurrentControlSet\Control\Nls\Language\  y  0412h - 1402`

  La figura 4.13 explica que existe un acceso en el código fuente a la clave de registro que almacena el idioma del sistema operativo. El nombre de la clave es *InstallLanguage* y el valor almacenado representa el código decimal y hexadecimal del idioma coreano.

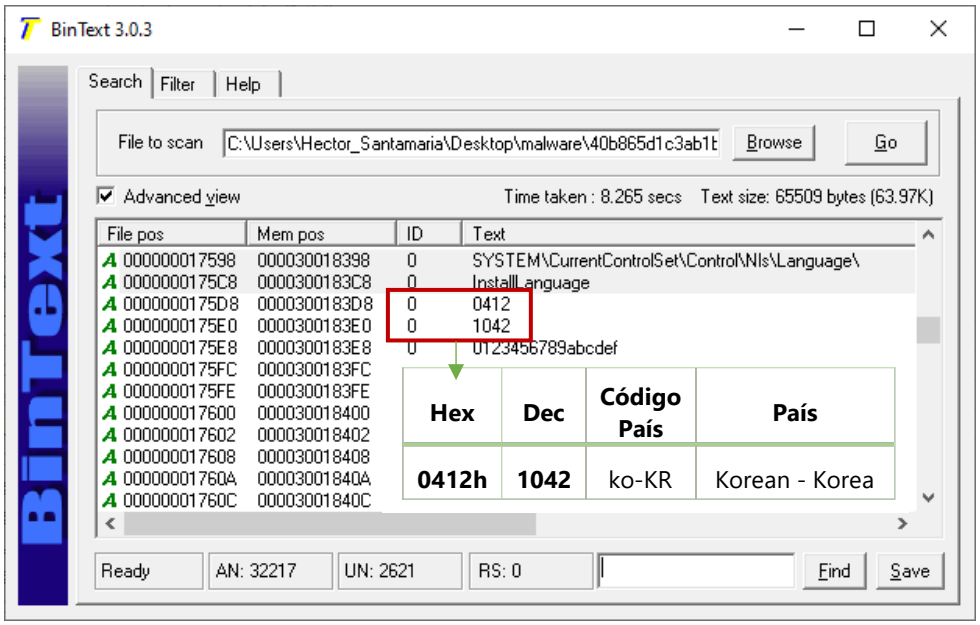


Figura 4.13 BinText mostrando los códigos de idioma y la clave de registro asociada recuperados del código.

Otras cadenas de texto que contienen todos los códigos de país existentes indican que se han utilizado funciones o librerías de lenguaje en el desarrollo del malware.

Es en esas cadenas de texto y la existencia de llamadas a ciertas librerías donde se han levantado varias sospechas sobre su modus operandi que serán confirmadas en los siguientes apartados, como, por ejemplo, si es capaz de pivotar entre otros elementos de la red, la forma en la que asegura su persistencia en el sistema operativo después de un reinicio o cómo evita que el usuario recupere sus datos desde puntos de restauración o copias de seguridad.

5. Análisis dinámico

En esta fase se pone en funcionamiento la muestra analizada estáticamente en el anterior apartado. La ejecución se realiza en un entorno controlado formado en una máquina virtual Windows conectada a otra Ubuntu que simula servicios de red, ambas con los adaptadores de red en modo *Host Only*.

En los experimentos de propagación y movimiento lateral se han utilizado dos máquinas Windows 10 en modo *Host Only* visibles entre ellas. La monitorización de lo que ocurre en el sistema operativo la efectúa *Process Hacker*, *Process Monitor* y *Noriben*. Para la red la captura de paquetes y su interpretación la proporciona WireShark.

Con el fin de clarificar la información presentada, se tomarán como referencia los mismos números de los procesos generados en una de las capturas de Process Monitor.

Quizá un punto que puede resultar confuso es que la muestra crea en los primeros momentos de actividad de una copia de sí mismo, luego la deposita en el mismo directorio del malware original para, acto seguido, iniciarla. El evento resultante es un proceso padre que toma el control de la copia como proceso hijo que puede verse en la figura 5.1. En caso de ejecutar la copia generada, ésta crearía otra réplica de sí misma y dependerá de la actividad de la instancia del binario que la haya creado.

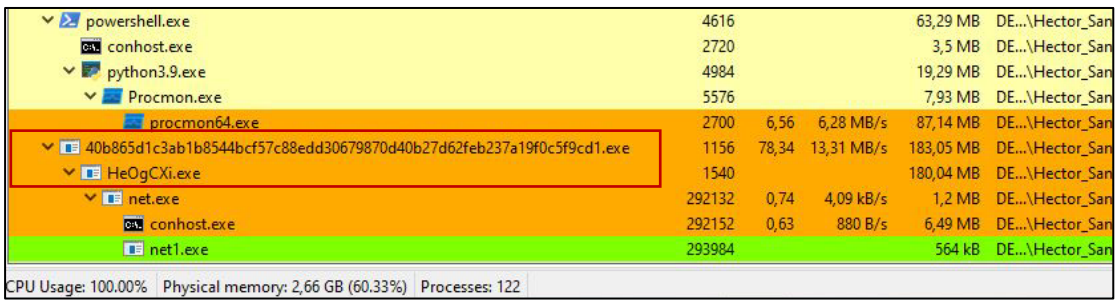


Figura 5.1 Detalle que muestra el proceso padre (ransomware original) y el proceso hijo (copia generada con nombre aleatorio).

5.1- Secuencia de acciones

La secuencia de acciones, cuyas implicaciones se detallarán en los siguientes apartados, es la siguiente:

1) Explorer.exe con PID-4700 arranca el proceso correspondiente al binario objeto de análisis gracias a la interacción del usuario.

2) Se crea el proceso malicioso Ryuk con PID-1156 con permisos de administrador bajo una arquitectura de 32-bit y este a su vez crea con éxito el hilo 2740.

3) Arranca el servicio de *Windows Program Compatibility Assistant (PCA)*
   a. El proceso PCA Crea un nuevo archivo *Windows Registry Hive Log*: *Amcache.hve*.

   b. Finalmente escribe varios valores de registro en las ubicaciones:
      i. *\REGISTRY\A\{ec9e6579-a869-b4e6-76a3-ddd5e0f91145}\Root\InventoryApplicationFile*

      ii. *REGISTRY\A\{ec9e6579-a869-b4e6-76a3-ddd5e0f91145}\Root\InventoryApplicationFile\40b865d1c3ab1b85|f90772eb51beb7bb\ProgramId*

4) El ransomware Ryuk PID-1156 a través del hilo 2740 crea y escribe en la ruta donde se alojaba el malware original una copia de sí mismo con un nombre aleatorio de siete caracteres en mayúsculas o minúsculas, correspondiente, en este caso, al binario *HeOgCXi.exe*. Realiza esta operación hasta en tres intentos.

5) El ransomware Ryuk PID-1156 a través del hilo 2740 en la ruta `HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap` escribe los siguientes valores de forma repetida en dos ocasiones:

   a. ProxyBypass
   b. IntranetName

c. UNCAsIntranet
d. AutoDetect

6) El ransomware Ryuk PID-1156 a través del hilo 2740 ejecuta el binario antes creado HeOgCxI.exe

7) Se crea el proceso malicioso *HeOgCxI.exe* PID-1540 con permisos de administrador bajo una arquitectura de 32-bit y este a su vez crea con éxito el hilo 4764 asociada a la copia escrita en el paso anterior. HeOgCxI.exe creará varios hilos.

8) El ransomware Ryuk PID-1156 a través de un nuevo hilo ejecuta la aplicación de Windows net.exe para 32-bit en la ruta *C:\Windows\SysWOW64\net.exe* con la línea de comando stop `"C:\Windows\System32\net.exe" stop "audioendpointbuilder"`

9) El proceso Net.exe abre Conhost.exe versión 32-bit en la ruta C:\Windows\System32\Conhost.exe con la línea de comando `\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1` generando un proceso nuevo y varios hilos. Realizará llamadas a Conhost.exe con el mismo comando de forma reiterada.

10) El proceso Net.exe ejecuta *Net1.exe* y éste crea un nuevo proceso y varios hilos.

11) El ransomware Ryuk PID-1156 mediante el hilo principal 2740 vuelve a lanzar net.exe y trata de detener un servicio con la línea de comando: "`C:\Windows\System32\net.exe" stop "samss" /y` para tratar de detener un servicio.

12) El ransomware Ryuk PID-1156 mediante el hilo principal 2740 desde net.exe trata de detener un servicio con la línea de comando: "`C:\Windows\system32\net1 stop "sdrsvc" /y`

13) El ransomware Ryuk PID-1156 a través de su hilo 3092 llama al proceso svchost.exe con el comando:

`C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted -p -s AudioEndpointBuilder`.

Eliminando además entradas y valores del registro en la ruta `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\MMDevices\Audio\Render\...`

14) El ransomware Ryuk PID-1156 a través de su hilo 1344 llama al proceso svchost.exe que ejecuta el comando `C:\Windows\system32\svchost.exe -k LocalService -p`

15) La copia HeOgCxI.exe PID-1540 mediante el hilo 4764 intenta acceder y cifrar un archivo concreto en la ruta: *C:\Users\Hector_Santamaria\AppData\Roaming\Microsoft\Crypto\RSA* con resultado fallido: *"NAME NOT FOUND"* al no encontrar el fichero.

16) La copia HeOgCxI.exe PID-1540 mediante el hilo 4764 crea un archivo en la ruta: *C:\ProgramData\Microsoft\Crypto\RSA\MachineKeys*

17) La copia HeOgCxI.exe PID-1540 mediante el hilo 4764 inicia icalcs.exe con el comando: `icacls "C:\*" /grant Everyone:F /T /C /Q` Este comando se ejecutará repetidamente

18) El proceso Icalcs.exe ejecuta conhost.exe mediante el comando: `\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1`

19) El ransomware Ryuk PID-1156 a través de su hilo 5992 llama a services.exe que inicia el proceso vssvc.exe *(Volume Shadow Copy Service).*

20) La copia HeOgCxI.exe PID-1540 mediante el hilo 4764 inicia la consola de comandos de Windows: *C:\Windows\SysWOW64\cmd.exe* y ejecuta el comando `cmd /c "WMIC.exe shadowcopy delet"`

21) La copia HeOgCxI.exe PID-1540 lee la tabla ARP y lanza peticiones *UDP Echo a su contenido*. Por ejemplo:

a. `DESKTOP-KAJJF8C:49667 -> 224.0.0.22:echo`
b. `DESKTOP-KAJJF8C:49667 -> 224.0.0.22:echo`
c. `DESKTOP-KAJJF8C:49668 -> 239.255.255.250:echo`
d. `DESKTOP-KAJJF8C:49668 -> 192.168.10.22:echo`

22) El ransomware Ryuk PID-1156 a través del hilo 2740 llama a la aplicación cmd.exe y escribe la siguiente clave de registro a través de un comando: `"C:\Windows\System32\cmd.exe" /C REG ADD "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "svchos" /t REG_SZ /d "C:\Users\Hector_Santamaria\Desktop\malware\40b865d1c3ab1b8544bcf57c88edd30679870d40b27d62feb237a19f0c5f9cd1.exe" /f /reg:64`

    Y unos segundos después otra clave con la dirección de la carga útil HeOgCxI.exe

23) Terminada esta última operación comienza el recorrido y cifrado de la mayoría de las carpetas en las unidades de almacenamiento.

5.2- Interacciones con el sistema de ficheros

La forma en la que interactúa este ransomware con el sistema de archivos se resume en su tratamiento de los archivos y carpetas, así como la forma que tiene de gestionar el recorrido, aunque también hay algunos aspectos sobre las creaciones de ficheros que conviene desarrollar.

5.2.1- Recorrido a través del sistema de archivos

Durante el cifrado de archivos, las principales operaciones mostradas por los programas de análisis son:

- CreateFile: Para crear las páginas web HTML con el mensaje y el correo electrónico de contacto. También es utilizado para abrir cualquier fichero.

- WireFile: Para escribir contenido en los archivos HTML anteriormente creados y cifrar cada elemento de la carpeta.

- SetRenameInformationFile: Para renombrar el archivo recientemente cifrado añadiendo la extensión .RYK

Una característica fundamental observada, además de notable, es que Ryuk crea un nuevo hilo por cada evento que perpetra en el sistema operativo y, del mismo modo, por cada fichero al que accede o crea, pudiendo así finalizar el cifrado de cada archivo con mucha rapidez.

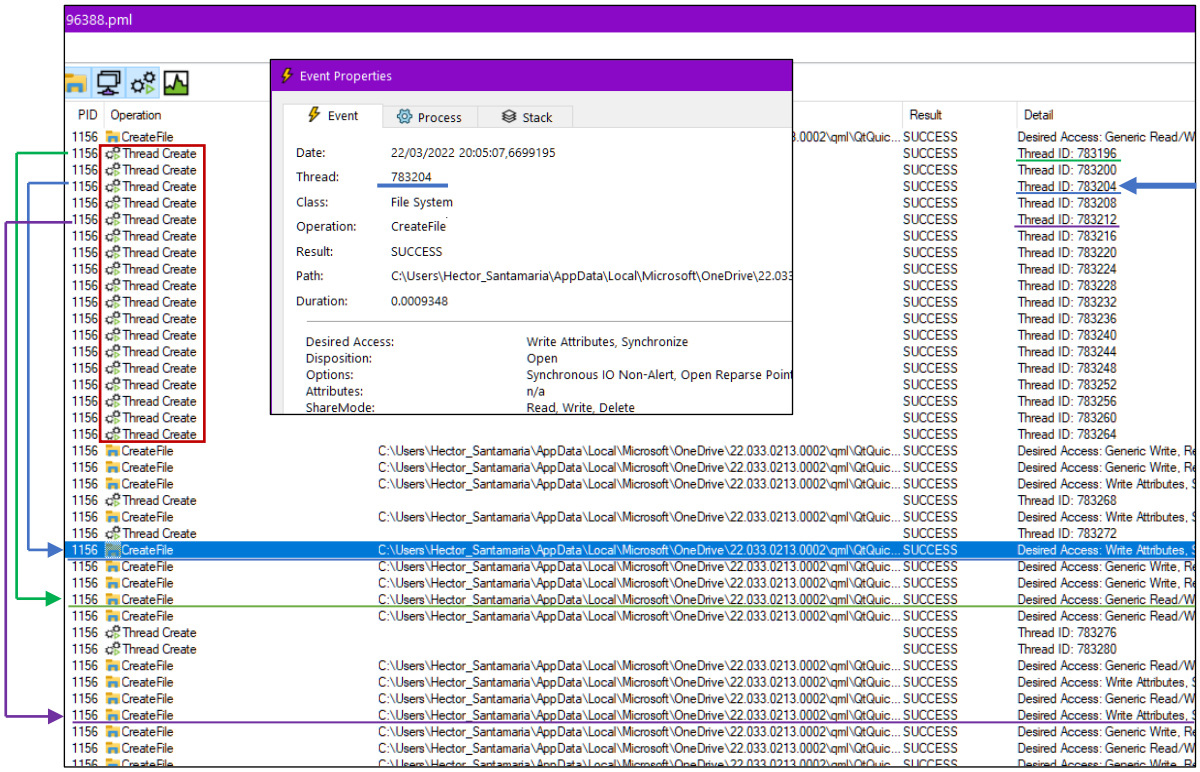


Figura 5.2 Captura de eventos de Process Monitor mostrando la relación entre hilos y eventos.

Como se observa en la figura 5.2, cada hilo tiene como objetivo la lectura, modificación o creación de cualquier archivo del sistema.

Puede saberse qué directorios y subdirectorios ha llegado a acceder, ya que en todos y cada uno de ellos genera un archivo “RyukReadMe” de extensión HTML cuyo contenido consta de una dirección de correo protonmail, el nombre del malware y un eslogan.

En el escritorio de cada usuario se depositan dos ficheros RyukReadMe.HTML para que en los reinicios el proceso malicioso, persistente gracias a las alteraciones en el registro de Windows, los abra automáticamente y se muestre a la víctima el mensaje prefijado. Esto último es posible por la existencia de al menos un navegador funcional, motivo por el que existe una excepción para las siguientes carpetas en sus métodos cifrado:

- Mozilla Firefox: *C:\Program Files\Mozilla Firefox* en la cual, no solo no cifra ningún archivo, sino que tampoco deposita el fichero de hipertexto.

- Actúa de la misma forma con Google Chrome en C:\Program Files\Google\Chrome\

- Papelera de reciclaje: Los elementos reciclados quedan indemnes.

- Carpeta Windows: Deposita sus característicos archivos HTML en cada subdirectorio, pero no altera ningún contenido.

Igualmente, el ransomware no encripta:

- Archivos de menos de 25 Bytes

- Archivos ejecutables (incluidos librerías *DLL* o .sys).

- Archivos de procesamiento por lotes. (.bat)

- Archivos de configuración (*.ini)*

- Archivos HTML

- Archivos .ico

Los ficheros deben tener, al menos, un tamaño igual o superior a 25 Bytes, Respecto al tamaño máximo, no ha podido determinarse; Ha podido cifrar archivos de hasta 10GB. Para realizar estas pruebas se generaron varios archivos *.txt* de diferentes tamaños rellenos de caracteres nulos con el comando:

```
fsutil file createnew prueba1.txt <<tamaño en bytes>>
```

La finalidad de estas acciones está relacionada por el interés de mantener el sistema operativo con un mínimo de funcionalidades, además de permitir acceder a internet o visualizar el correo especificado dentro de RyukReadMe.html con un navegador ya que, una vez acabado el proceso de encriptado, es imposible ejecutar algún otro a excepción del obsoleto Internet Explorer y muchas de las aplicaciones del sistema quedan inservibles.

En la figura 5.3 pueden apreciarse los efectos de Ryuk en una carpeta aleatoria con herramientas para el análisis dinámico. También como Microsoft Edge ha quedado totalmente

inservible, mientras que el directorio de Mozilla Firefox no ha sido alterado. El mensaje en HTML que se incluye es similar en todas las rutas del sistema.

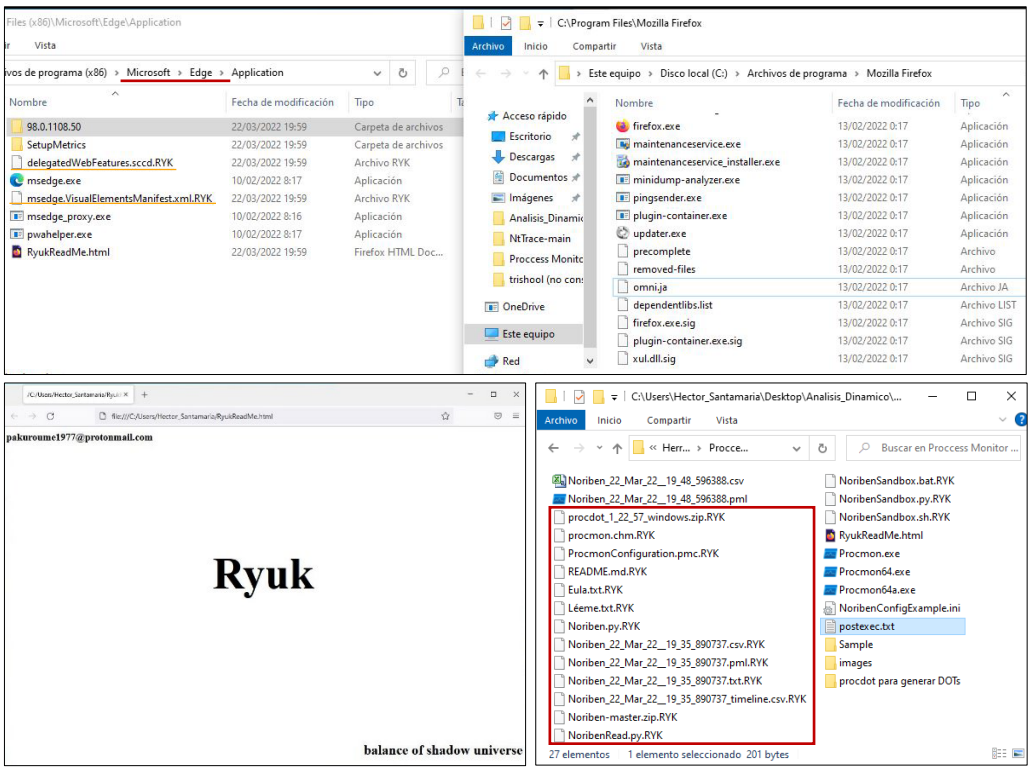


Figura 5.3 Detalles del estado de los archivos y carpetas del sistema operativo. En la parte inferior izquierda el repetido mensaje HTML dejado por Ryuk.

5.2.2- Borrado de copias de seguridad

El resto de las interacciones con el sistema de ficheros consiste en el borrado de las Shadow Copies y, aunque es cierto que teóricamente el ransomware dispone de dos métodos para su eliminación, en las pruebas realizadas se pudo observar que solamente utilizó el comando **`WMIC.exe`** `shadowcopy delet` si la funcionalidad de copia de seguridad y restauración no se encontraba activada en Windows. De cualquier modo, Ryuk sí inicia el proceso `C:\Windows\system32\vssvc.exe` en ambos casos. Si las copias de seguridad están habilitadas ejecuta posteriormente **`vssadmin.exe`** `Delete Shadows /all /quiet`, si no, empleará el recurso WMI para el borrado.

La utilidad de línea de comandos para acceder al Windows Management Instrumentation (WMIC.exe) es la infraestructura para la administración operaciones en Windows y solo puede ser utilizada por los administradores del sistema. (Documentación oficial de Microsoft, 2021).

Únicamente, con esos parámetros, el ransomware solo puede borrar una copia de seguridad a la vez, ya que Windows puede preguntar confirmación a través de la consola de comandos, motivo por el cual tal vez cree una gran cantidad de hilos para eliminarlas por medio de WMIC.exe mientras transcurre su ejecución, como ejemplifica la figura 5.4. No obstante, se desconoce por qué el comando está incompleto, pero nunca llega a tener efecto porque falta la "e" final en "delet".

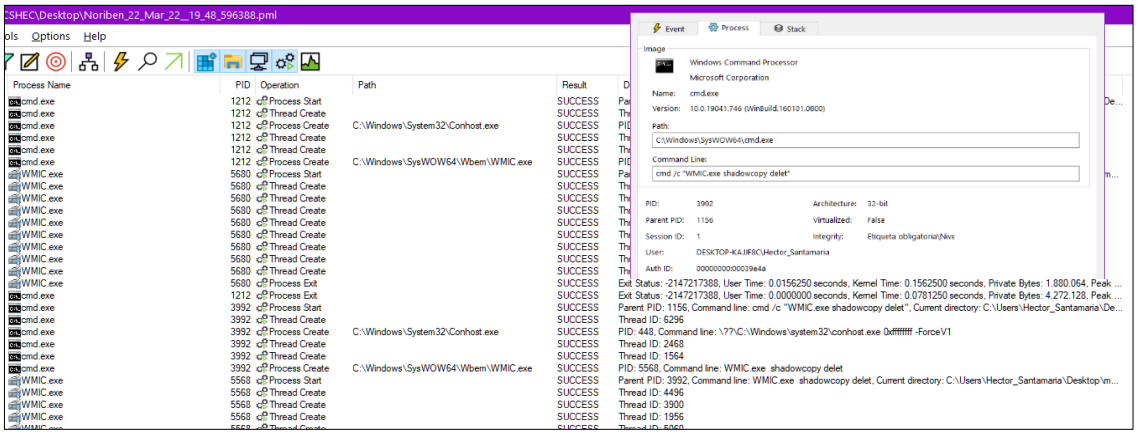


Figura 5.4 Filtrado de eventos con múltiples hilos para la ejecución del comando shadowcopy delete.

5.2.3- Estadísticas sobre el proceso de cifrado

En la fase en la que comienza el cifrado de archivos y el recorrido por los directorios del sistema, el uso del procesador alcanza el 100% al igual que aumenta la ocupación de memoria RAM. El impacto en el rendimiento del sistema se ve afectado independientemente de utilizar mayor o menor cantidad de núcleos en la CPU. La diferencia estará en el tiempo transcurrido hasta la finalización.

Completando todo lo posible la información que se puede obtener del proceso de encriptado en el sistema de archivos, mediante los datos recogidos por Process Monitor y del

propio sistema operativo atacado, se han generado una serie de estadísticas que se recogen en la tabla 5.1 y 5.2:

Datos sobre carpetas y archivos existentes (Excluyendo carpeta Windows)	
Número total de directorios	19728
Número total de archivos	170241
Tamaño total (Megabytes)	24094821
Tamaño promedio de archivo (Megabytes)	0,1415334

Tabla 5.1 Número de carpetas y archivos existentes en la máquina infectada.

Estos son los datos sobre los archivos, directorios y subdirectorios existentes en la unidad de almacenamiento. Como referencia se puede tomar un sistema operativo recientemente instalado, con actualizaciones, programas para uso ofimático (Microsoft Office, Adobe Reader) navegación (Mozilla Firefox, Google Chrome) herramientas de desarrollo (Visual Studio 2022, Notepad++) y todas las utilidades implicadas en el análisis estático, dinámico y de código (Python, Wireshark, PE-Bear, x64dbg, Ghidra, etc.).

En cuanto al tiempo de cifrado medio por cada archivo y finalización total del proceso:

Datos sobre tiempos de cifrado y duración del proceso	
Duración promedio de cifrado por archivo (WriteFile)	0,00034247 (s)
Duración total de todas las operaciones de Cifrado (WriteFile)	59,0420550 (s)
Duración total de todas las acciones maliciosas:	1 hora, 45 minutos

Tabla 5.2 Tiempos promedios de cifrado y la duración total.

No se contemplan los tiempos de acceso a cada carpeta y creación o lectura de ficheros. Concretamente, el acceso en milisegundos a un archivo es difícil de medir ya que la operación *CreateFile* se aplica a la creación de los ficheros de hipertexto HTML con el mensaje de los autores y al acceso de cualquier archivo potencialmente encriptable. Aunque tampoco se mide el tiempo de acceso a cada carpeta, es fácil hacerse a la idea según el tiempo total de la actividad de Ryuk y las características de la máquina virtual empleada.

5.3- Interacciones con el sistema operativo

Los cambios y accesos a recursos en el sistema operativo observados son interesantes desde el punto de vista del movimiento lateral y la persistencia.

En el momento del arranque, se pone en marcha el servicio *Program Compatibility Assistant (PCA)*, lanzado por *svchost.exe*. Como ya se ha explicado, las características de la muestra analizada revelaban que la versión mayor del sistema operativo había sido establecida para Windows XP, por lo que PCA actúa en consecuencia y escribe algunos valores en el registro de Windows para evitar conflictos y fallos del sistema. Se trata de una acción que no tiene mayor importancia, pero que puede conllevar cierta confusión durante el análisis.

5.3.1- Cambios en las opciones de internet

La primera interacción directa tiene lugar en los valores de las claves de registro ubicadas en …\`Internet Settings\ZoneMap` con el objetivo de facilitar sus operaciones en la red local. De hecho, la figura 5.5 evidencia que Ryuk provoca estos cambios dos veces seguidas, en dos intentos:

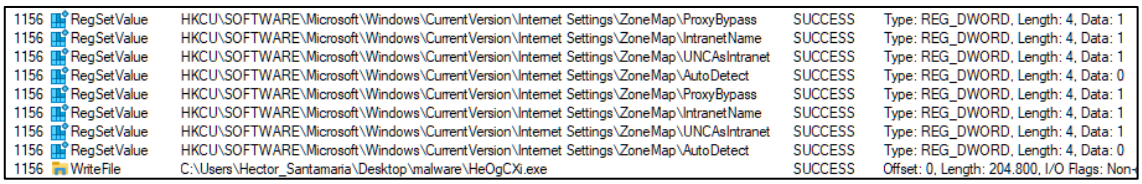
- Permite que los sitios web u otros elementos de la red que omitan el servidor proxy instalado (en el caso de haberlo) se puedan asignar a la zona de intranet local con `Settings\ZoneMap\ProxyByPass : 1`

- Habilita que todas las direcciones de red y sitios locales se consideren siempre dentro de la zona de seguridad de intranet porque, si no estuviera así especificado, la zona de seguridad de aplicación sería la de internet. Además, evita que sean los propios usuarios los que deban escoger en qué zona debería encontrarse: `Settings\ZoneMap\IntranetName: 1`

- Establece que todas las direcciones de red (Rutas UNC) se asignen a zona de intranet. Una ruta UNC se utiliza para el acceso a los recursos de la red, como puede ser una unidad de almacenamiento remota o una carpeta de un servidor, siempre que se

antepongan las dos barras invertidas. Por ejemplo: *PC_Principal**CarpetaCompartida1*. La forma de activarlo es con: `Settings\ZoneMap\UNCAsIntranet: 1`

- Desactiva la detección automática del tipo de intranet y la aplicación de las reglas asociadas a esa intranet se aplicarán sin importar cómo estén configuradas en el caso de que la computadora pertenezca a un dominio determinado, es decir, evita configuraciones automáticas si forma parte de una red bajo administración: `Settings\ZoneMap\AutoDetect: 0`



1156	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
1156	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
1156	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
1156	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect	SUCCESS	Type: REG_DWORD, Length: 4, Data: 0
1156	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
1156	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
1156	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
1156	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect	SUCCESS	Type: REG_DWORD, Length: 4, Data: 0
1156	WriteFile	C:\Users\Hector_Santamaria\Desktop\malware\HeOgCXi.exe	SUCCESS	Offset: 0, Length: 204.800, I/O Flags: Non-

Figura 5.5 Entradas de registro de opciones de internet modificadas con sus correspondientes valores.

La alteración del registro de Windows en este sentido facilita, creando el escenario ideal, la detección y el acceso a otros dispositivos en la red local. Si bien esta configuración es más habitual para un usuario doméstico, no lo sería tanto en una red corporativa.

5.3.2- Afianzar la persistencia en el sistema

El único método observado para persistir después de un reinicio del sistema operativo es insertando una entrada en el registro de Windows desde la línea de comandos en una de las cuatro claves *CurrentVersionRun*, concretamente:

```
"HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"  /v
"svchos" /t REG_SZ /d.
```

Nótese que los autores han escogido la entrada *run* y no *runonce* para que no se elimine en posteriores inicios de sesión y han camuflado el programa de inicio como "*svchos*", sin la 't', quizá con el fin de simular que se trata del típico proceso *svchost.exe*.

La figura 5.6 muestra como desde la herramienta Autoruns de Sysinternals se visualiza la nueva entrada como programa autoejecutable, además del comando exacto a través de *cmd.exe* que se empleó para escribirla.

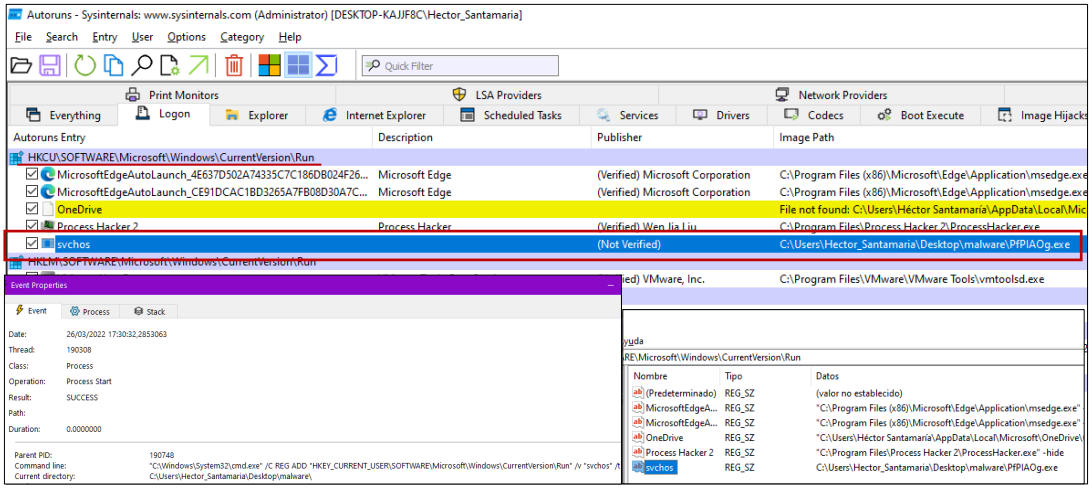


Figura 5.6 Clave de registro añadida en CurrentVersion\Run para arrancar la carga útil después de reinicio.

La ruta que figura como valor de la nueva clave en el registro no es siempre la ubicación de la copia, aunque ésta se genera siempre dentro de la ruta donde se encontraba el archivo malicioso principal. Ryuk, habiendo transcurrido muy poco tiempo desde su inicio, crea la clave y establece su valor apuntando hacia el binario original y, en una segunda ronda, sobrescribe el valor de esa misma clave con la ruta y nombre del binario copia para asegurarse de que prospera la infección si por algún motivo no ha podido escribirla.

5.3.3- Servicios de Windows alterados

Entre las acciones de la muestra analizada y algo que viene siendo habitual en todas las variantes y registros de infección de este ransomware, está desactivar el servicio de sonido de Windows. Hasta la fecha, ningún analista ha sabido determinar cuál es la razón de este proceder.

Algunas empresas de seguridad y analistas creen que la finalidad es aumentar la percepción del usuario de que su computadora no funciona bien y provocar un malestar adicional. (G-Data Security Lab, 2019).

Primeramente, detiene el servicio de audio con `stop "audioendpointbuilder"` a través de *net.exe* y en un segundo tiempo elimina entradas y valores del registro en `…\CurrentVersion\ MMDevices\Audio\Render\...` para deshabilitarlo definitivamente perdurando en posteriores reinicios mediante un proceso *svchost.exe* con el comando `-k LocalSystemNetworkRestricted -p -s AudioEndpointBuilder s`

Otro servicio de Windows que también desactiva el código dañino es Windows Security Accounts Manager (samSs). Este servicio administra las cuentas de usuarios y grupos del sistema operativo. Ryuk se sirve de net1.exe para detenerlo y samSs está ligado a los siguientes servicios relevantes:

- Remote Procedure Call (RPC): Utilizado para solicitar servicios y realizar llamadas a procesos ubicados en computadoras remotas en la red.

- DCOM Server Process Launcher: También asociado a la comunicación entre sistemas en una red. Un cliente permite ejecutar remotamente componentes de software distribuidos (DCOM) en otro equipo Windows.

- RPC Endpoint Mapper: Permite asignar números de puertos a servicios RPC con el fin de facilitar la comunicación.

En las investigaciones realizadas por el alumno, los mencionados servicios SamSs, rpcSs y DcomLaunch seguían funcionando con normalidad, samSs no pudo ser desactivado y, por lo tanto, RPC, DCOM y RPC Endpoint Mapper no se vieron afectados. De cualquier modo, como aparece expuesto en la captura de pantalla de la figura 5.7, en el hipotético caso de que sí lo estuviera el Administrador de Cuentas de Seguridad: SamSs, los demás servicios relacionados sí seguirían funcionando y esto tiene una utilidad, porque se trata de recursos de los que este ransomware se vale para sus transmisiones de código y réplicas hacia otras máquinas de la

red local y la desactivación de samSs es útil para que las soluciones SIEM no puedan auditar correctamente los sistemas operativos Windows y no reciban alertas de lo que está ocurriendo.

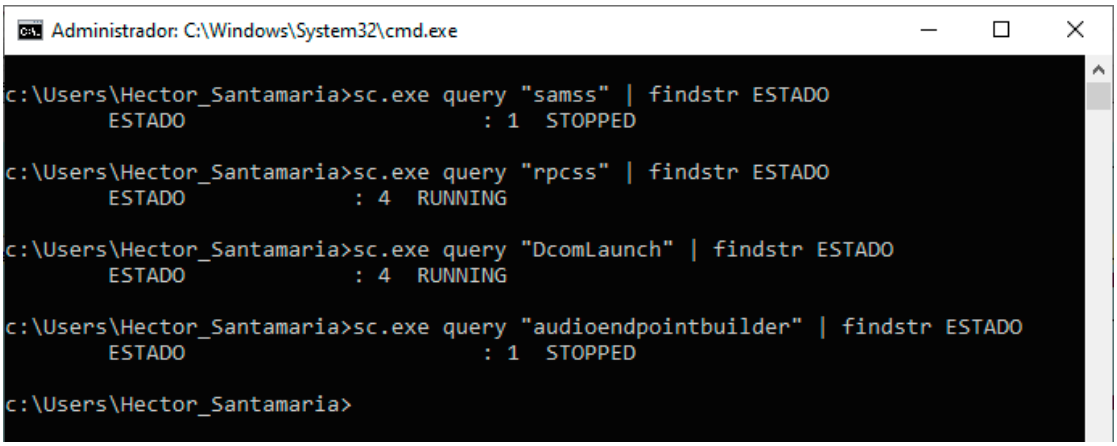


Figura 5.7 Estado de samss detenido forzosamente junto con el servicio de audio finalizado por Ryuk. Los servicios RPC y DCOM funcionan con normalidad.

La última interacción que Ryuk tiene con los servicios de Windows es también para detener sdrsvc (Microsoft Windows Backup Service) que ofrece las funcionalidades de copias de seguridad y restauración de Windows: `C:\Windows\system32\net1 stop "sdrsvc"`.

De nuevo recurre a Microsoft Windows Networking: Net.exe y a la versión con funcionalidades extendidas Net1.exe para variar el estado de los servicios, aunque esta consola le permitiría llevar a cabo operaciones con usuarios, grupos o conexiones de red, por ejemplo.

5.3.4- Otras características modificadas

Ya se había explicado en este documento como la cadena de texto sospechosa `icacls "C:\*" /grant Everyone:F /T /C /Q` podría ser utilizada para añadir todo tipo de permisos a los contenidos de las unidades de almacenamiento y, efectivamente, el programa malicioso ejecuta esta acción de forma insistente.

Icalcs.exe así como net1.exe se apoyan en procesos conhost.exe (Console Window Host) que se sitúa en medio del ClientServer Runtime System Service (CSRSS) y la consola del sistema CMD.exe y mediante la instrucción `conhost.exe 0xffffffff -ForceV1` solicita información directamente desde el espacio del Kernel, por lo que lee del espacio de la memoria del sistema que comprende del (0x80000000-0xffffffff) en una arquitectura de 32-bit

Si se examinan las cadenas de caracteres dentro del volcado de memoria de un proceso Conhost cualquiera desde Process Hacker, pueden verse los comandos y parte del contenido descubiertos en el análisis estático pues, realmente, está bajo el control del binario Ryuk ejecutado para el análisis, lo cual se verifica en la figura 5.8.

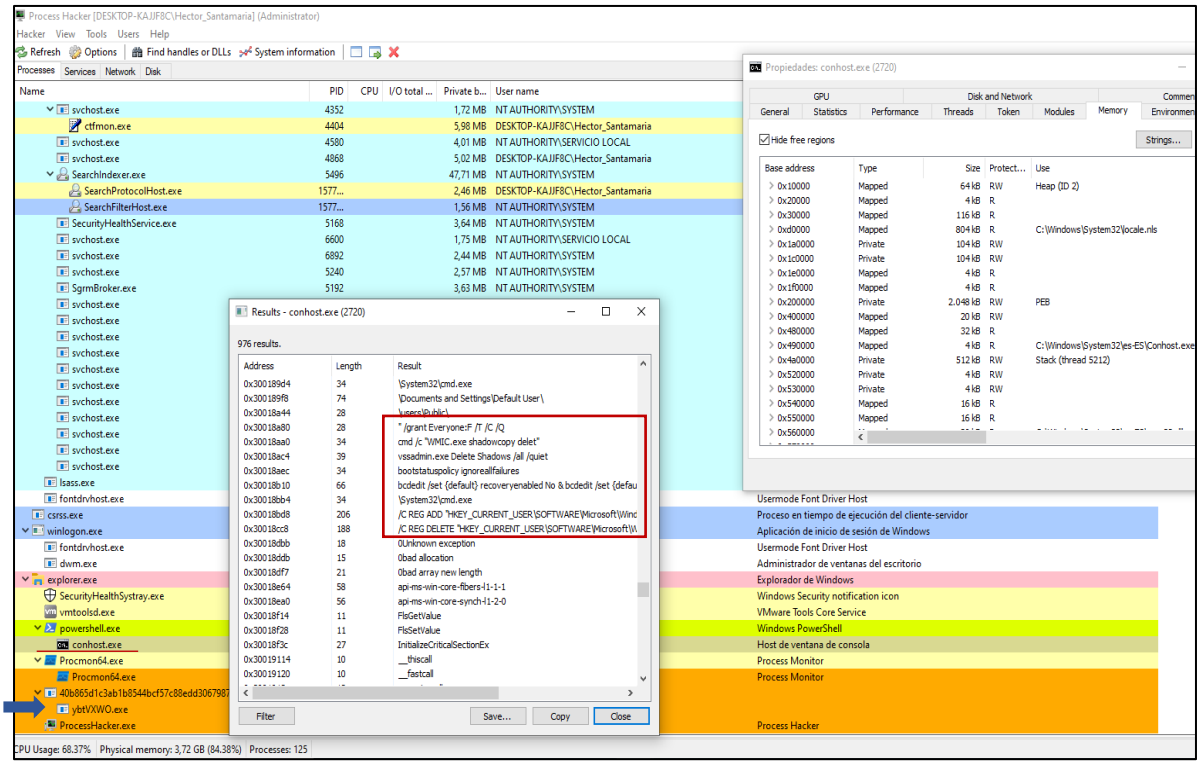


Figura 5.8 Strings del proceso conhost.exe invocado por el binario copia que Ryuk crea al inicio de su ejecución, señalada con flecha azul.

5.3.5- Funciones no detectadas

A pesar de los indicios que presentaban los comandos, llamadas a programas y otros textos hallados en el código de la muestra Ryuk que se está analizando, su implementación, tal y como se esperaba, finalmente no se ha llevado a cabo.

Según los datos del análisis estático, teóricamente Ryuk debería interactuar con el BCD lanzando un comando `bcdedit /set recoveryenabled No` y `bcdedit /set bootstatuspolicy ignoreallfailures`. Examinando los registros sysmon, de procmon o consultando directamente la configuración desde *bcdedit.exe*, nada corrobora ningún intento de modificación de la configuración de inicio. En la figura 5.9 se aprecia que los parámetros BCD se muestran igual antes y después de la ejecución del programa malicioso:

```
C:\Windows\system32>bcdedit.exe

Administrador de arranque de Windows
------------------------------------
Identificador           {bootmgr}
device                  partition=\Device\HarddiskVolume1
path                    \EFI\Microsoft\Boot\bootmgfw.efi
description             Windows Boot Manager
locale                  es-ES
inherit                 {globalsettings}
default                 {current}
resumeobject            {b555eb68-7550-11eb-8020-ee1304ca5b8b}
displayorder            {current}
toolsdisplayorder       {memdiag}
timeout                 30

Cargador de arranque de Windows
-------------------------------
Identificador           {current}
device                  partition=C:
path                    \Windows\system32\winload.efi
description             Windows 10
locale                  es-ES
inherit                 {bootloadersettings}
recoverysequence        {b555eb6a-7550-11eb-8020-ee1304ca5b8b}
displaymessageoverride  Recovery
recoveryenabled         Yes
isolatedcontext         Yes
allowedinmemorysettings 0x15000075
osdevice                partition=C:
systemroot              \Windows
resumeobject            {b555eb68-7550-11eb-8020-ee1304ca5b8b}
nx                      OptIn
bootmenupolicy          Standard
bootstatuspolicy        DisplayAllFailures

C:\Windows\system32>net user

Cuentas de usuario de \\HECTOR-1

-------------------------------------------------------------------------------
Administrador            DefaultAccount           Hector_Santamaria
```

Figura 5.9 Configuración de BCD después de un ataque con Ryuk.

De igual modo, otros *strings* representaban los códigos de idioma para sistemas operativos correspondientes a Corea, junto con la clave de registro que los contenía. Se modificó totalmente el idioma de las máquinas virtuales al del país asiático, pero el comportamiento de Ryuk fue idéntico al de otras ocasiones. Un ejemplo de la realización de estos cambios en el sistema operativo y el nulo efecto que tuvieron hacia el avance del ransomware está representado en las capturas de pantalla de la figura 5.10, donde el reporte de Noriben demuestra que la actividad en el sistema operativo, en cuanto a creación de procesos, tráfico de red y entradas en el registro de Windows, era igual que en ejecuciones anteriores.

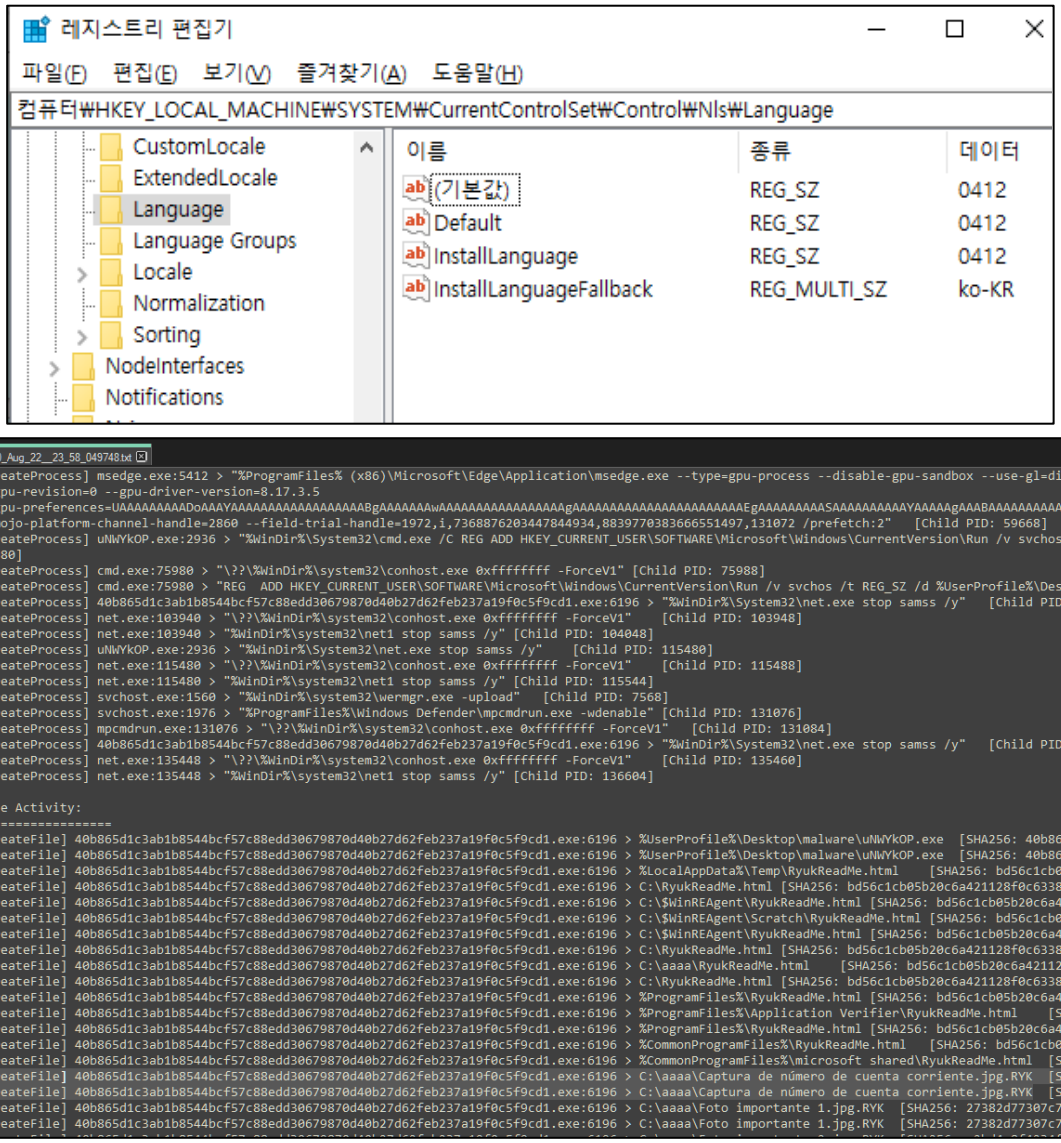


Figura 5.10 Claves de registro establecidas en idioma coreano y reporte de actividad habitual en Ryuk

5.4- Capacidades de red y movimiento lateral

La muestra analizada es capaz de propagarse a través de la red local hacia otras máquinas Windows. En este apartado se describirá con detalle el proceso en el entorno de pruebas desde una máquina Windows a otra. Con el fin de dejar clara la diferencia entre los dos hosts con los que se trabajan y su rol, en adelante:

- El nombre de la máquina Windows propagadora será: HECTOR-1
- El nombre de la segunda máquina en la red receptora del ransomware será: HECTOR-2

5.4.1- Servicios y recursos utilizados

Uno de los primeros indicios durante el análisis dinámico ocurre en los primeros segundos de la infección cuando ordenadamente ejecuta:

- *Wmiprvse.exe* desde un hilo creado por proceso malicioso principal con los parámetros `-secured -Embedding`. Permite lanzar un archivo ejecutable de forma remota a través WMI e interactuar con otros dispositivos remotos o locales conectados a la máquina infectada.

- Un proceso *svchost.exe*, que aparece representado en la figura 5.11, cuyo hilo padre fue creado por el proceso malicioso principal con la instrucción `-k NetworkService -p -s Dnscache`. El propósito es doble:

    - Primero, mediante Dnscache trata de encontrar direcciones IP asociadas nombres de host en la red local cuando no hay un servidor DNS que proporcione resolución de nombres, lo hay y no los proporciona o todos los intentos hallan fallado. La resolución de nombres de los dispositivos conectados funciona a través de este servicio a partir de Windows Vista o siguientes versiones. (Stanek. W, 2014)

    - Segundo, con NetworkService puede crear y manejar conexiones a cualquier dirección IP local.

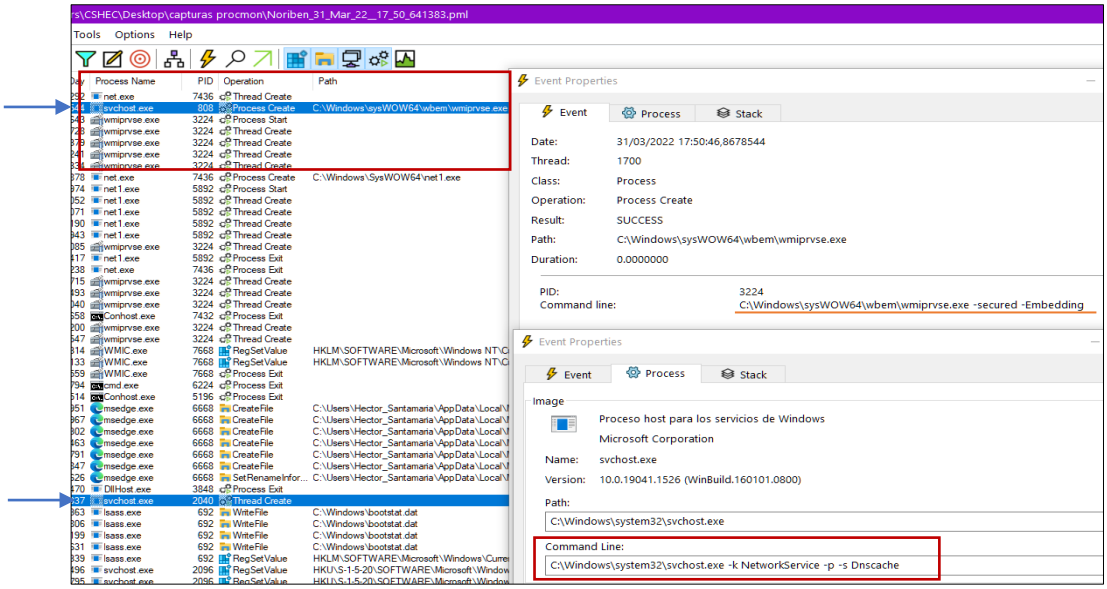


Figura 5.11 Wmiprvse.exe, NetworkService y DnsService lanzados por Service Host (svchost.exe).

Respecto al protocolo de comunicaciones con el que opera, es SMB. Se ha podido verificar como no existe ningún impedimento para extender la infección si la versión SMBv2 y SMBv3 está desactivada en alguno de los hosts, pero sí se mantiene en funcionamiento SMBv1 o viceversa. Lo crucial es que cualquiera de las dos versiones del protocolo esté activa. La figura 5.12 plasma ese comportamiento.

Actualmente Windows 10 no incorpora SMBv1 y requiere de instalación manual. Si no se hallase ninguna de las tres versiones SMB habilitada, Ryuk no podría moverse entre hosts aunque haya disponible un recurso compartido en la red sin restricciones de acceso.

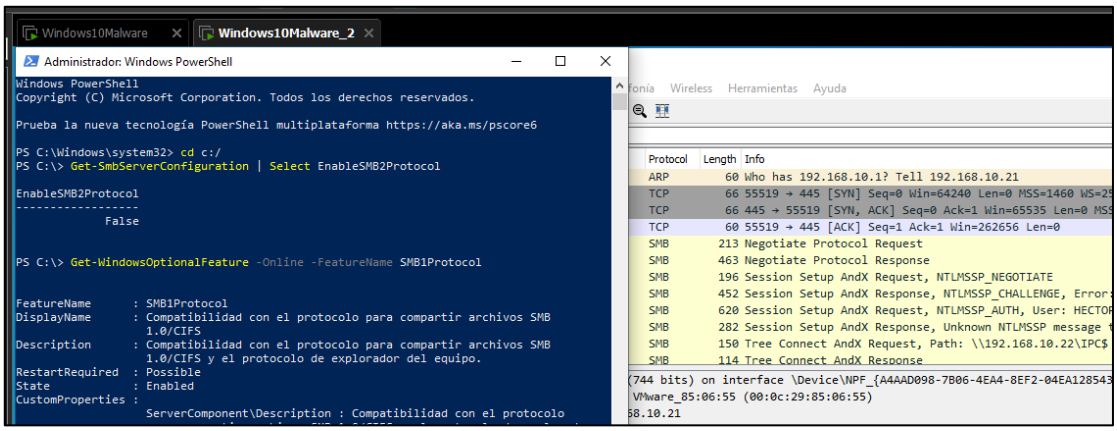


Figura 5.12 Conexión creada por Ryuk a través de SMBv1. PowerShell muestra como SMBv2 y SMBv3 se encuentran desactivados.

5.4.2- Detección de otros dispositivos conectados

Ryuk necesita, en primer lugar, detectar dispositivos conectados en la red para comenzar su propagación. Todo el proceso relativo al movimiento lateral y funciones de red lo realiza el binario malicioso copiado al comienzo de la ejecución del malware principal. Utilizará el parámetro 8 LAN en esta línea de comando: `malware\nOVVOAm.exe" 8 LAN` Donde *nOVVOAm.exe* es el nombre aleatorio que atribuye a la copia. Cuando Ryuk detecta un PC remoto al que puede conectarse, ese binario auto replicado se dedica exclusivamente a cifrar sus archivos, volviendo a operar sobre la computadora originaria una vez haya finalizado.

En los primeros momentos de la ejecución encuentra la dirección IP del host conectado y utiliza el protocolo NetBios para enviar una petición "Name Query" descubriendo así el nombre tras dicha dirección: HECTOR-2. Acto seguido genera paquetes Wake-on-LAN para reactivar computadoras en estado de suspensión o hibernación a través de UDP hacia todas las direcciones que contenga la tabla ARP del sistema operativo. El contenido de cada uno de esos paquetes consta del valor 255 en hexadecimal junto con dieciséis repeticiones de la dirección MAC (Media Access Control) del host detectado:

```
ffffffffffff000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e000c29a16d7e
```

Inmediatamente después transmite solicitudes Echo también para las IPs detectadas en ARP. En el caso del sistema operativo Windows contendrá normalmente:

- Dos Direcciones multicast de clase D dentro del rango Local Network Control Block asociadas a protocolos de enrutamiento o descubrimiento de topologías a bajo nivel por el IANA:

  - **224.0.0.22**: Reservada al protocolo IGMP que permite descubrir hosts dentro de grupos en transmisiones multidifusión.

  - **224.0.0.251**: Reservada al protocolo Mdns (multicast DNS). mDNS se utiliza en redes de pequeño tamaño donde existen varios dispositivos que requieren resolución de nombres para su conexión, pero sin disponer de un servidor DNS tradicional. La

petición para encontrar un host en la red se realiza por multidifusión por lo que cuando un dispositivo coincide con la consulta devuelve su dirección IP.

- Una dirección de uso privado no reservada y que puede utilizar cualquier asociación: **239.255.255.250**

También existe una entrada a la dirección IP de la máquina virtual Windows que se ha configurado para recibir el malware a través de la red local: **192.168.10.22**.

Las direcciones de multidifusión podrían considerarse un factor de riesgo como medio útil para detectar dispositivos conectados de forma masiva a través de los paquetes Echo. La petición Echo enviada a 192.168.10.22 deriva en una respuesta Echo y se produce un ping a través de ICMP en la que se verifica la disponibilidad de la máquina HECTOR-2. En la figura 5.13 se pueden observar estas peticiones que emplean la API *IcmpEchoRequest*, la respuesta ping generada y el tráfico del intercambio de nombres con NetBios.

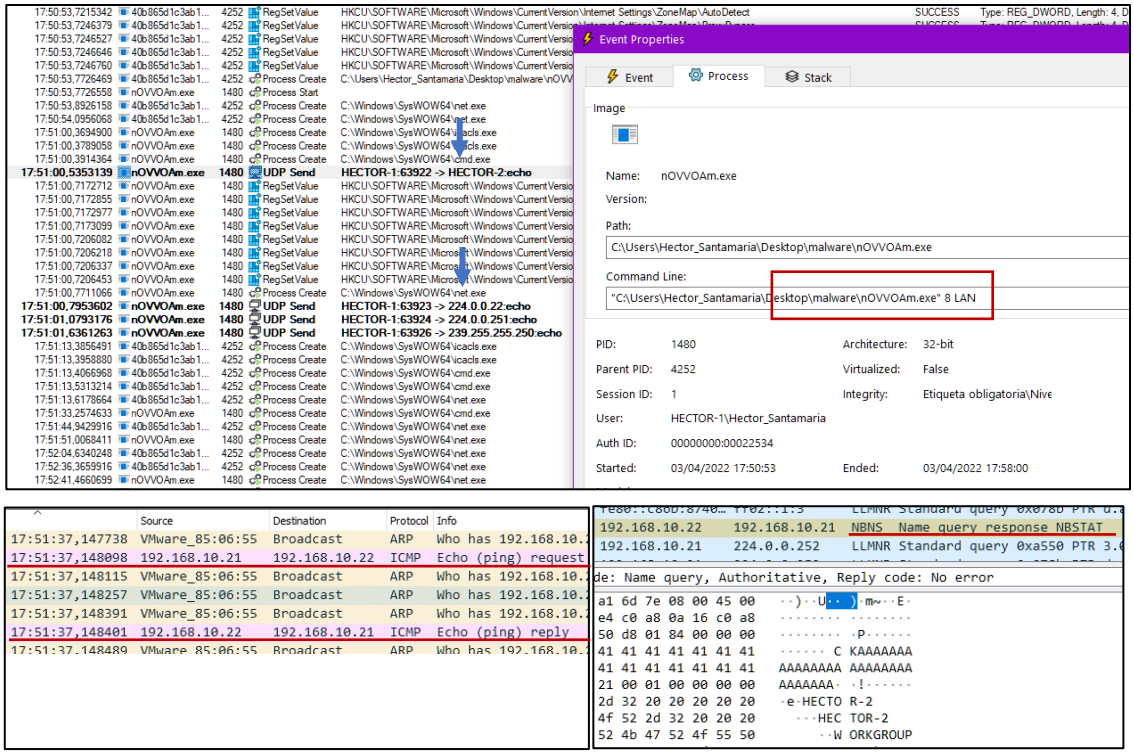


Figura 5.13 Solicitudes UDP ECHO capturadas por Process Monitor y tráfico asociado monitorizado por Wireshark.

5.4.3- Movimiento Lateral en hosts dentro de un dominio

Para estudiar el comportamiento de Ryuk entre computadoras Windows dentro de un dominio, fue sustituida la máquina virtual Ubuntu Desktop LTS por una versión Ubuntu Server con la misma configuración de red. Sobre Ubuntu Server se crea el dominio *HRANSOM.COM* con Samba Active Directory, Kerberos y Winbind. La figura 5.14 muestra algunos detalles de la configuración de ese servidor.

Figura 5.14 Configuración de servidor Samba en Ubuntu

Las máquinas virtuales Windows se incluyen en el dominio y, además, se crea un recurso compartido en el servidor de forma que sea visible entre ambas, pues de otra forma es imposible que Ryuk establezca una conexión para infectar y cifrar los archivos de otros ordenadores en la red. Si no hay una carpeta o unidad de almacenamiento entre la computadora infectada y la computadora bajo amenaza, no podrá transmitirse.

Se crean así dos usuarios para la conexión al dominio: *hector_ryuk* y *hector2_ryuk2*, más la carpeta compartida *shared,* apreciables en la figura 5.15, tanto en el inicio de sesión en Windows 10 junto con el correspondiente dominio, como en el listado de usuarios que se puede pedir desde la consola de comandos de Ubuntu.

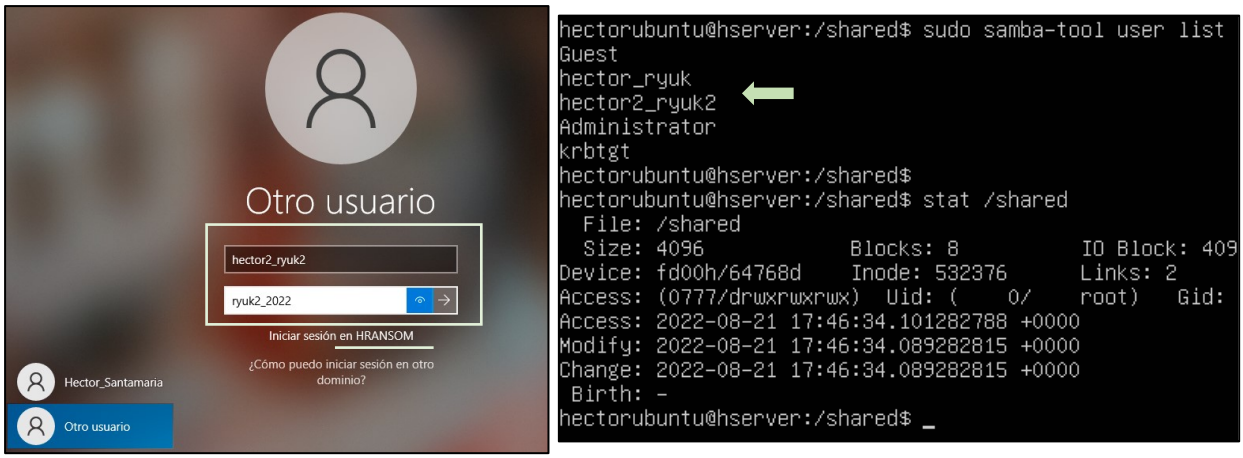


Figura 5.15 Inicio de sesión en Windows para usuarios creados en Samba AD Server

Al ejecutar el ransomware este se propaga encontrando la carpeta compartida y comenzando a cifrar uno a uno todos los archivos de las unidades de almacenamiento remotas siguiendo la misma metodología que para los ficheros locales . Únicamente es necesario que la carpeta compartida esté montada como ubicación de red en el resto de las computadoras; si la ubicación no estuviera agregada en la máquina foco de la infección, igualmente podría seguir pivotando entre otras que cumplan el requisito.

Cuando detecta que se encuentra en una computadora remota, crea un proceso para replicarse a través de SMB en la ubicación descubierta en el análisis de cadenas de texto sobre las interacciones con el sistema de archivos: *C:\Users\Public\ finish*. Esta copia no supondría ninguna amenaza de no ejecutarse, sin embargo, Ryuk emplea el programa instalado con Windows, Schtasks.exe sobre el cuál Microsoft detalla que *"Permite a un administrador crear, eliminar, consultar, cambiar, ejecutar y finalizar tareas programadas en un equipo local o remoto"*. (Documentación oficial de Microsoft, 2022).

Una vez ha transmitido el binario malicioso a las computadoras de la red, inicia Schtasks con el comando:

```
schtasks.exe /Create /S 192.168.10.22 /TN <nombre de tarea> /TR "C:\Users\Public\<nombre binario copiado>" /SC once /ST 00:00 /RL HIGHEST
```

El significado de cada uno de los parámetros es:

- **Create**: Crea una nueva tarea en el sistema operativo.

- **S** (system): Indica el equipo remoto al que se conectará. En este caso, es la dirección IP de la máquina HECTOR-2.

- **TN** (taskname): Nombre que identificará la tarea. El ransomware genera 7 caracteres aleatorios en mayúsculas y minúsculas para ello.

- **TR** (taskrun): Especifica cuál será la ruta completa y archivo de programa o tarea que se iniciará.

- **SC** (Schedule): Debe indicarse la frecuencia de ejecución. En este caso, los desarrolladores han optado por una sola vez.

- **ST** (start time): Hora programada para el inicio de la tarea. Son parámetros obligatorios y el malware establece por defecto las 12 horas de la madrugada.

- **RL** (level): El valor *Highest* ejecutará la tarea con permisos de administrador.

Después, lanza una segunda orden con la que ejecutará en ese mismo momento la tarea programada con el comando:

```
schtasks.exe /S 192.168.10.22 /Run /TN <nombre de tarea>
```

Se hace referencia al nombre de la tarea ya creada en el comando anterior y la pone en marcha con el parámetro *"Run"*. Para ejecutar la tarea, dentro de un dominio creado con Samba Active Directory, debe poseer los privilegios suficientes. La figura 5.16 contiene el tráfico de red capturado con WireShark durante este proceso.

A partir de este punto, el procedimiento de cifrado y demás interacciones con el sistema operativo son los habituales, pues el ransomware recién descargado al equipo remoto actúa de la misma forma.

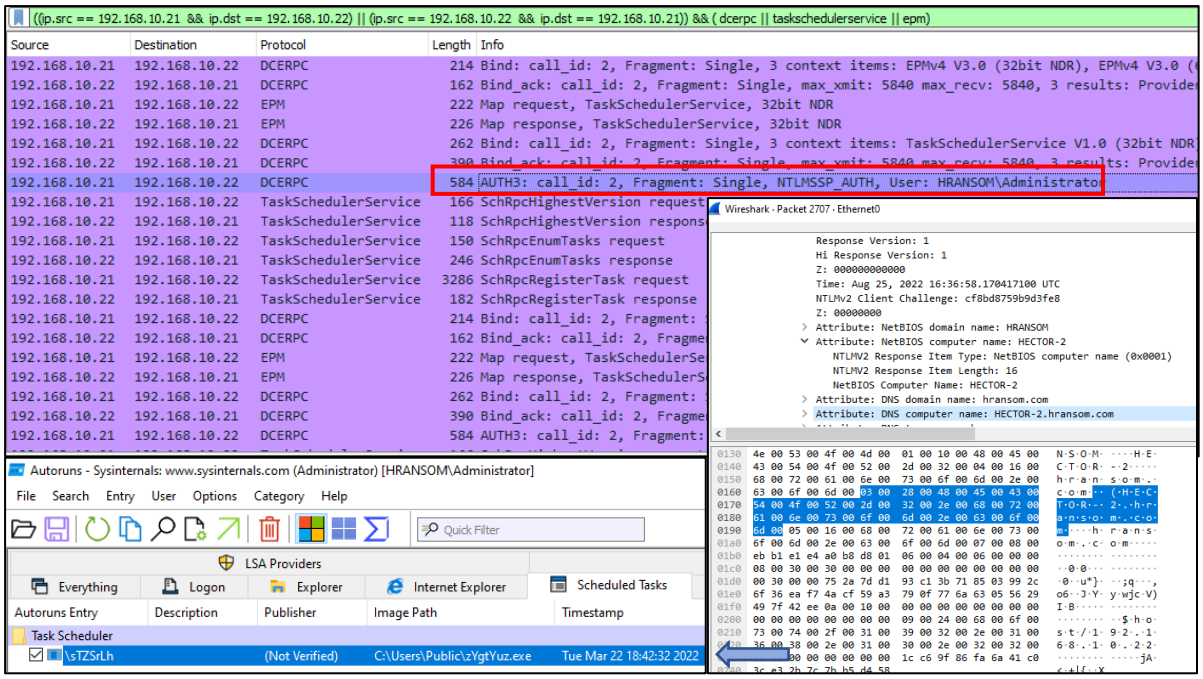


Figura 5.16 Tráfico RCP generado por Ryuk para crear la tarea de forma remota.

En la figura 5.17 se aprecia como el equipo que ha transmitido el ransomware recorre los directorios del host remoto desde un proceso *system*, mientras se inicia la nueva instancia de Ryuk a través de *svchost.exe*, que hará lo mismo y, a su vez, buscará nuevos objetivos en la red.

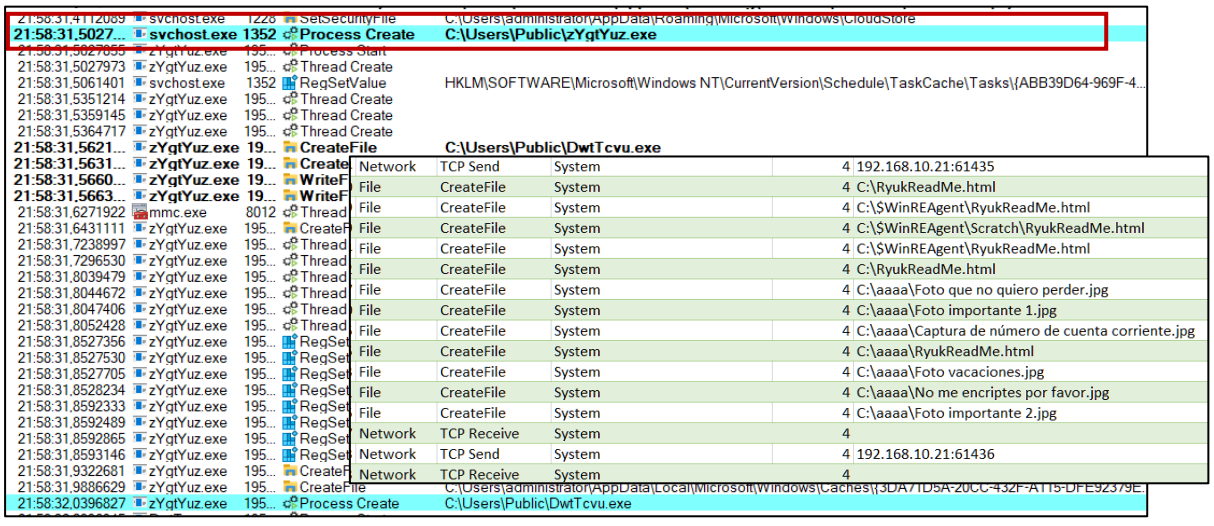


Figura 5.17 Se inicia una nueva instancia de Ryuk desde una tarea programada mientras la instancia original cifra los archivos de forma remota

5.4.4- Limitaciones en la propagación

Aunque una de las características de Ryuk es comportarse como un gusano infectando otros equipos conectados a la red local, se han podido observar ciertas limitaciones cuando se trata de sistemas operativos modernos como Windows 10 en sus últimas versiones: 2004, 20H2, 21H1 o 21H2. Prácticamente todas ellas convergen en dos puntos en común, las políticas de Windows para detectar recursos compartidos y el protocolo SMB.

Durante los análisis quedó demostrada la inviabilidad de que, en dispositivos Windows 10 fuera de un dominio, el malware pueda establecer conexiones entre dos equipos de una red sin que se lleven a cabo configuraciones concretas y muy poco seguras para proporciar acceso a sus unidades de disco.

5.4.4.1- Dos o más hosts sin contraseña de usuario fuera de un dominio

Para este caso, se deben llevar a cabo ciertas configuraciones que permitan al código dañino extenderse y cifrar archivos en computadoras remotas.

Siguiendo con la nomenclatura utilizada en el apartado sobre el movimiento lateral, la máquina receptora HECTOR-2 necesita disponer de al menos las siguientes modificaciones en el sistema operativo:

- Activado el uso compartido de archivos e impresoras.
- Desactivado el uso compartido con protección por contraseña.

Estos cambios permiten que se establezca la comunicación, sin embargo, la conexión no llega a efectuarse devolviendo HECTOR-2 el error: *STATUS_ACCOUNT_RESTRICTION*. El siguiente paso consiste en habilitar inicios de sesión de invitado no seguros en HECTOR-1 a través de una directiva de grupo local "*Habilitar inicios de sesión de invitado no seguros*" para *Estación de Trabajo Lanman*. Esto es debido a que no se puede proporcionar acceso autenticado y se debe indicar al cliente SMB si admitirá estas conexiones vulnerables. Únicamente se encuentra deshabilitado el acceso de invitado en Windows 10 Enterprise y Windows 10 Education, pero sí es permitido en Windows 10 Home y Pro. Se ha de recordar

que la versión del sistema operativo en las máquinas virtuales es Windows 10 Enterprise 19041 y esta configuración se aplica desde la versión 1709

En la figura 5.18 queda patente que con las modificaciones realizadas la sesión se establece, pero Ryuk no consigue acceso a ningún fichero.

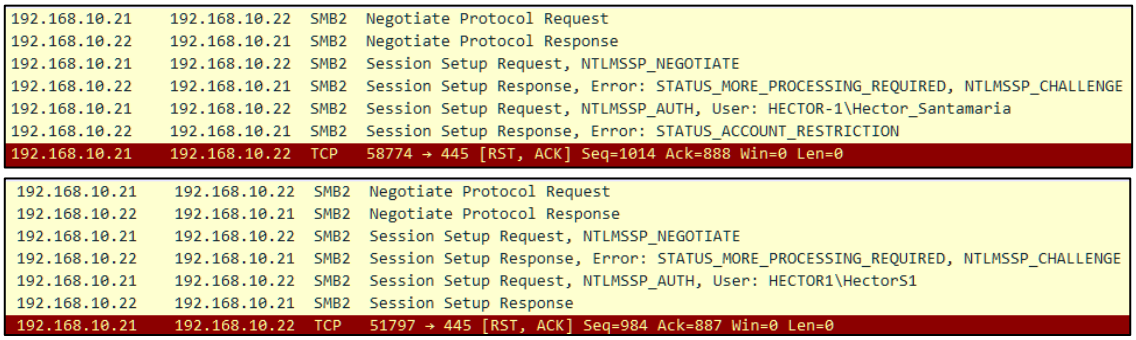


Figura 5.18 Arriba, la sesión no se establece con el error STATUS_ACCOUNT_RESTRICTION. Abajo, la sesión sí se establece al habilitar los inicios de sesión no seguros.

Para hacer totalmente factible el proceso de infección de la muestra, deben eliminarse algunas restricciones más en el sistema operativo de Microsoft. Otra de ellas es disponer del "LocalAccountTokenFilterPolicy" como clave de registro en el sistema con valor 0 (no habilitado) y así permitir la ejecución de comandos SMB en dispositivos en línea. Debe hacerse para la máquina a la que se desea transmitir el malware: HECTOR-2

La última limitación para el supuesto contexto actual está relacionada nuevamente con la carencia de autenticación en los usuarios locales de Windows. La directiva de seguridad local "*Limitar el uso de cuentas locales con contraseña en blanco solo para iniciar sesión en la consola*" obstaculiza el acceso a las unidades de las computadoras remotas carentes de contraseña (contraseña en blanco).

En la figura 5.19 se puede observar el evento de Windows generado por SMBServer y recopilado por Sysmon que se produce si no se desactiva la limitación de cuentas locales con contraseña en blanco en HECTOR-2, junto con el mensaje de acceso denegado al llegar Ryuk a la unidad "*C:*" visible en la red.

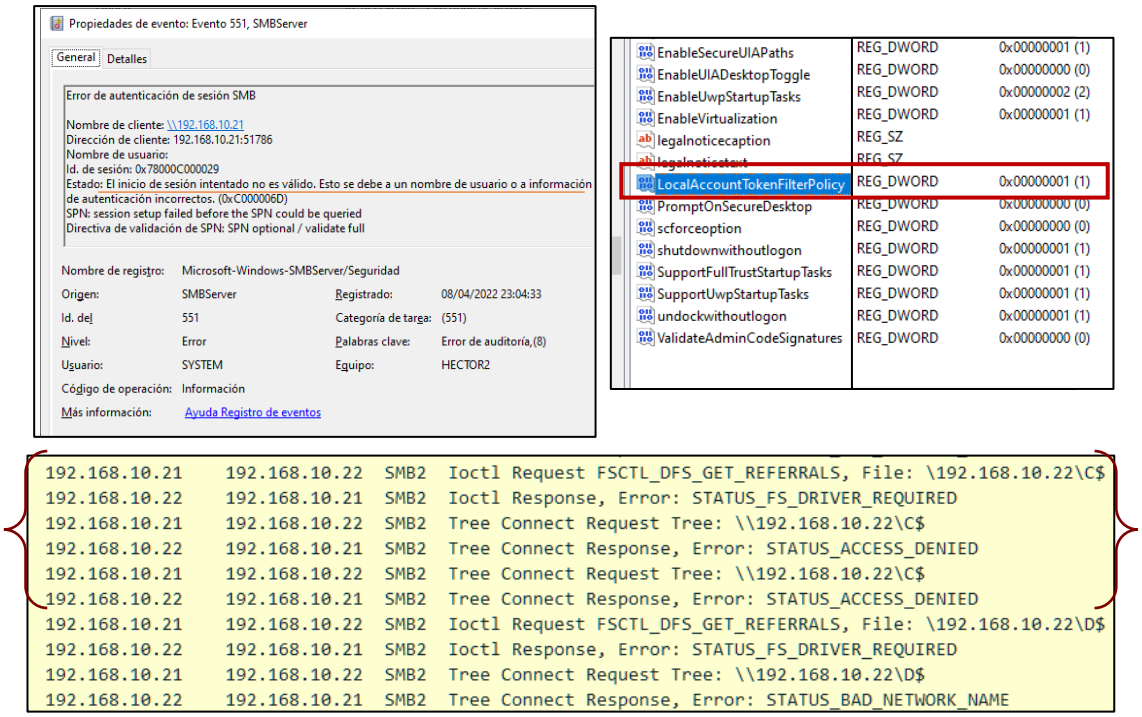


Figura 5.19 Evento de Windows y captura del tráfico de red SMB2 generado por Ryuk cuando LocalAccountTokenFilterPolicy se ha deshabilitado, pero sigue vigente la limitación de cuentas con contraseña en blanco.

Una vez acatados estos requisitos, ya es posible que Ryuk pivote entre otros equipos y unidades compartidas estando los hosts en un dominio o no.

5.4.4.2- Dos o más hosts con contraseña de usuario dentro y fuera de un dominio

Analizando otros supuestos, se comprobó como cuando sí figura una contraseña entre los usuarios de las computadoras únicamente se requiere la desactivación de *LocalAccountTokenFilterPolicy* en el host remoto que será infectado.

Según la documentación oficial (Microsoft, 2021), la finalidad de la restricción es impedir que un administrador local establezca una conexión remota con permisos elevados. Se trata, pues, de un Control de Cuentas de Usuario (UAC) remoto. El control de cuentas remoto no afecta a equipos dentro de un dominio gestionado por Active Directory, por lo que, como se mostrará más adelante, en un entorno empresarial Ryuk podría propagarse con cierta flexibilidad.

A pesar de ser uno de los requisitos para facilitar la transmisión del código malicioso del apartado anterior, no es necesaria ninguna otra modificación. Además, este filtro para tokens con permisos administrativos afecta a cuentas locales, pero no a cuentas dentro de un dominio. En la figura 5.20 se visualiza el tráfico de red con la conexión exitosa y una captura de Process Monitor donde el proceso maliciosa cifra archivos remotos en la dirección IP de HECTOR-2.

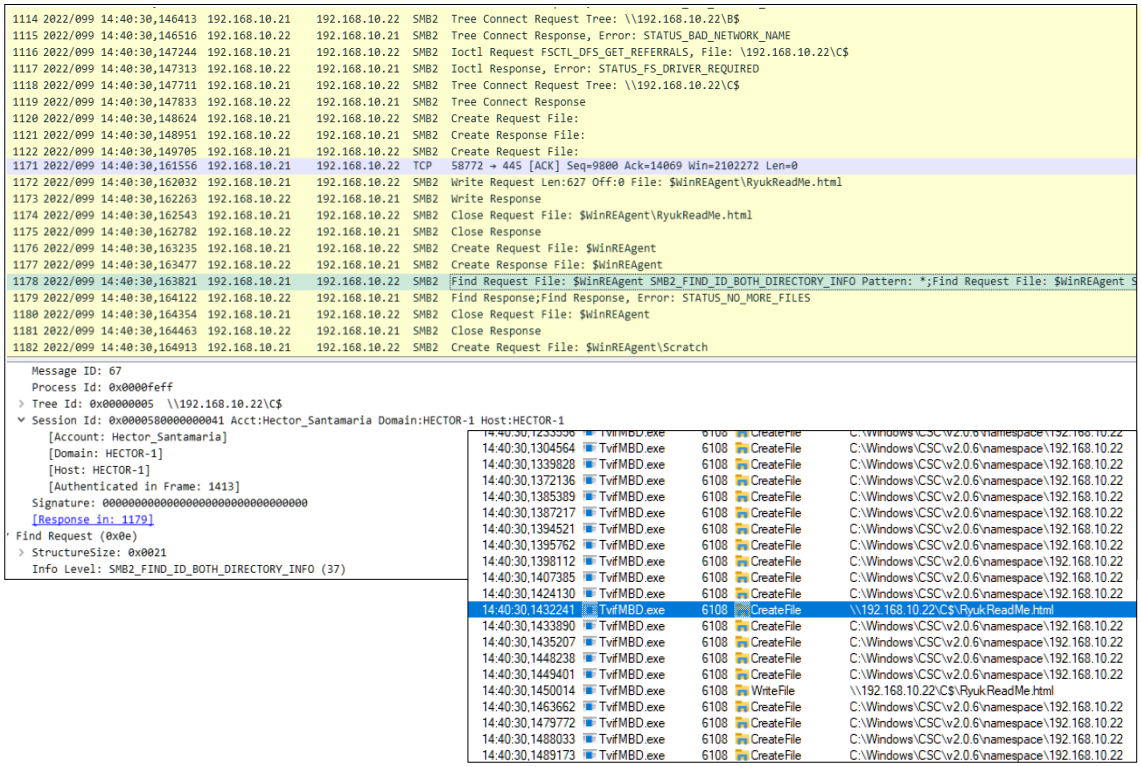


Figura 5.20 Detalle de la conexión establecida al equipo remoto junto con el comienzo de encriptado de sus archivos una vez establecido LocalAccountTokenFilterPolicy en HECTOR-2.

5.4.4.3- Direcciones IP públicas

Ryuk o al menos la compilación objeto de este trabajo, no está interesado en extenderse hacia máquinas con IP públicas ya que, si se configuran las direcciones IP de una o varias con direcciones públicas dentro del mismo rango, de forma que sea posible la comunicación, son ignoradas completamente.

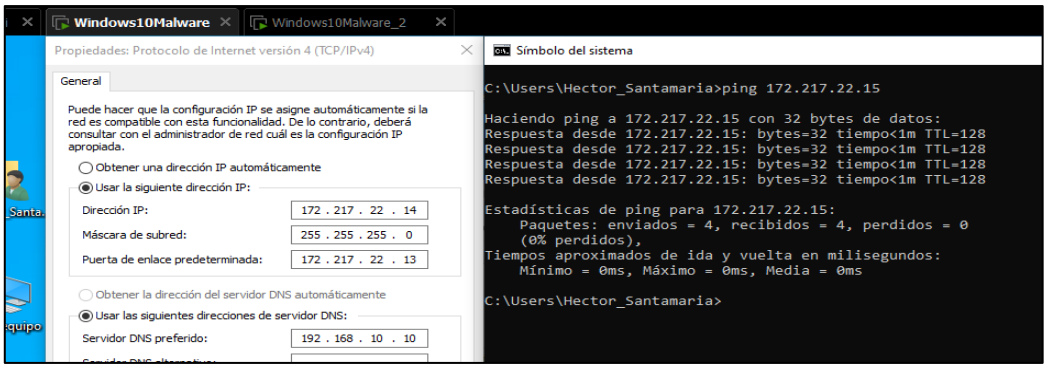


Figura 5.21 Configuración de ambas máquinas virtuales con direcciones IP públicas del mismo rango.

La figura 5.21 refleja que se utilizan las direcciones 172.217.22.14 y 172.217.22.15 para HECTOR-1 y HECTOR-2, posteriormente tratará de reactivar los hosts con un Magic Packet (Trama de difusión de 6 bytes con valor 255). Al ejecutar Ryuk, puesto que 172.217.22.15 se encuentra en la tabla ARP de HECTOR-1, Ryuk sí envía un paquete UDP wake-on-lan a HECTOR-2, sin embargo, no realiza ningún ping a través de ICMP ni cualquier otra acción posterior. Esto ocurre ejecutándolo en ambas máquinas conectadas a un posible objetivo remoto a infectar, donde nunca llega a establecer comunicación de algún tipo a excepción del paquete ECHO – UDP inicial, como puede comprobarse también en la figura 5.22.

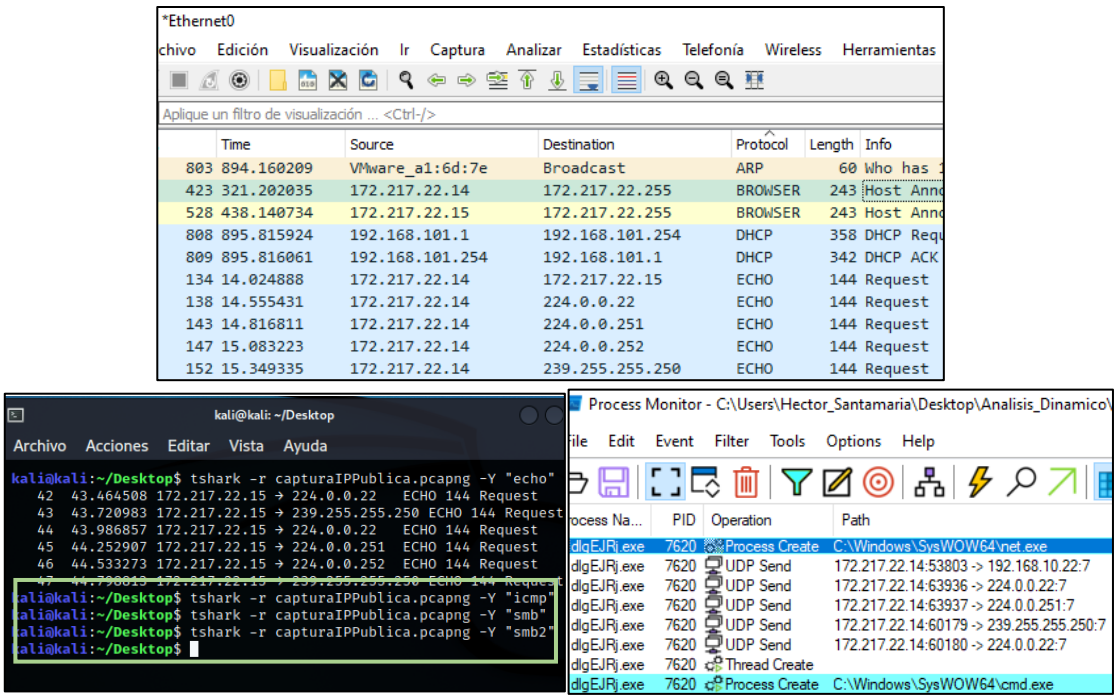


Figura 5.22 Un paquete UDP (ECHO) es enviado, no obstante, se descarta cualquier otra acción posterior al poseer los hosts IPs públicas.

5.5- Análisis de código

El análisis de código se realiza con el depurador open-source X64dbg en su instancia de 32 bits. El desarrollo de este apartado se centrará en las características más notables de Ryuk que, mayoritariamente, tienen que ver con las llamadas a funciones de API detectadas en el análisis estático o vistas en el dinámico.

5.5.1- Ofuscación de código y llamadas a API

En lo expuesto en el capítulo de análisis estático, resulta evidente que no hay ninguna referencia a ninguna función criptográfica y solamente suposiciones sobre las posibilidades que podría conferir al malware integrar ciertas DLL. La depuración del código reveló que las llamadas a las funciones criptográficas están ofuscadas. Los autores optaron por ofuscar dichas llamadas cargando y resolviendo los punteros de las funciones de API dinámicamente. Los punteros se resuelven llamando a *GetProcAddress* y, una vez obtenido el método de la función, ésta ya estaría disponible para utilizar, además de mostrarse en la sección de llamadas a módulos de X64dbg. Es un método alternativo en lugar de utilizar el enlace dinámico de tiempo de carga (Load-Time Dynamic Linking), que implicaría la aparición de los nombres de las APIs en la IAT (Import Address Table).

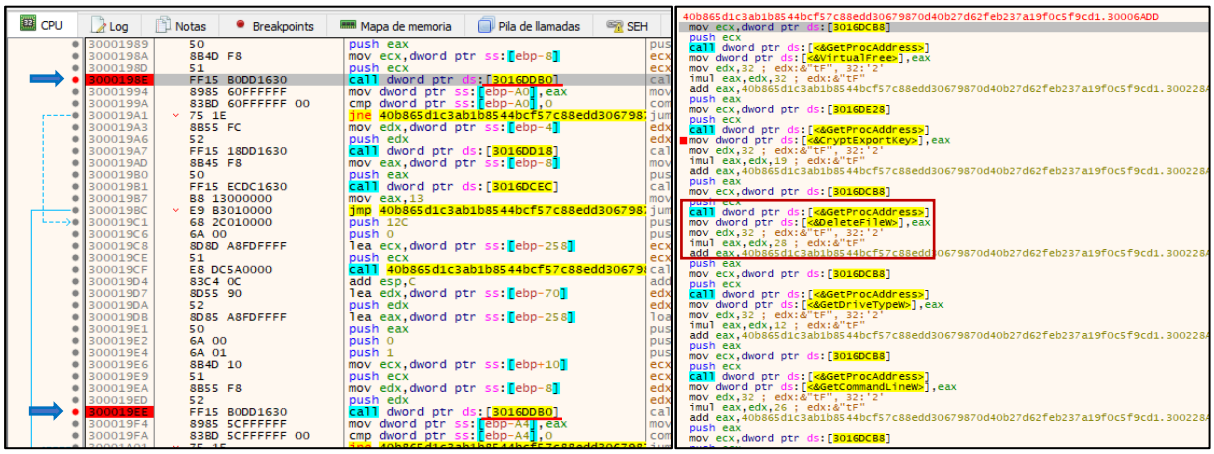


Figura 5.23 A la izquierda la función CryptExporKey nombreada por su referencia: 3016DD80. A la derecha, el método que resuelve todos los nombres con GetProcAddress.

Al inicio de la ejecución de Ryuk una llamada a un método resuelve todas las referencias secuencialmente. La figura 5.23 muestra el código ensamblador de ese método y a *GetProcAddress* resolviendo la referencia 3016DCB8 que después se asigna en la instrucción `mov dword ptr ds: [<&DeleteFilew>]`. Como particularidad, en la llamada a *GetIpNetTable* de la librería *Iphlpapi* han optado por ocultarla de la forma que aparece en la figura 5.24, en texto en claro dentro del código. Después de leerse, se almacena en el registro *eax* con la instrucción `push eax`. La llamada tampoco quedaría representada directamente en la tabla de importaciones, pero sí puede detectarse en un análisis manual de las cadenas de texto literales en el código fuente.

```
30006C3D  51              push ecx
30006C3E  8B15 B8DC1630   mov edx,dword ptr ds:[3016DCB8]
30006C44  52              push edx
30006C45  FF15 DC800130   call dword ptr ds:[<&GetProcAddress>]
EIP ->
30006C4B  A3 E8DD1630     mov dword ptr ds:[3016DDE8],eax
30006C50  C645 84 47      mov byte ptr ss:[ebp-7C],47      47:'G'
30006C54  C645 85 65      mov byte ptr ss:[ebp-7B],65      65:'e'
30006C58  C645 86 74      mov byte ptr ss:[ebp-7A],74      74:'t'
30006C5C  C645 87 49      mov byte ptr ss:[ebp-79],49      49:'I'
30006C60  C645 88 70      mov byte ptr ss:[ebp-78],70      70:'p'
30006C64  C645 89 4E      mov byte ptr ss:[ebp-77],4E      4E:'N'
30006C68  C645 8A 65      mov byte ptr ss:[ebp-76],65      65:'e'
30006C6C  C645 8B 74      mov byte ptr ss:[ebp-75],74      74:'t'
30006C70  C645 8C 54      mov byte ptr ss:[ebp-74],54      54:'T'
30006C74  C645 8D 61      mov byte ptr ss:[ebp-73],61      61:'a'
30006C78  C645 8E 62      mov byte ptr ss:[ebp-72],62      62:'b'
30006C7C  C645 8F 6C      mov byte ptr ss:[ebp-71],6C      6C:'l'
30006C80  C645 90 65      mov byte ptr ss:[ebp-70],65      65:'e'
30006C84  C645 91 00      mov byte ptr ss:[ebp-6F],0
30006C88  8D45 84         lea eax,dword ptr ss:[ebp-7C]
30006C8B  50              push eax
30006C8C  8B0D 20DD1630   mov ecx,dword ptr ds:[3016DD20]
30006C92  51              push ecx
30006C93  FF15 DC800130   call dword ptr ds:[<&GetProcAddress>]
30006C99  A3 44DD1630     mov dword ptr ds:[3016DD44],eax
30006C9E  BA 32000000     mov edx,32                       32:'2'
```

Figura 5.24 Función GetIpNetTable expresado en caracteres hexadecimales

Las dos largas cadenas de texto cifradas halladas en el análisis estático, contenían el texto del archivo HTML que se genera como aviso en cada directorio. En la figura 5.26 se puede ver que, por cada fichero generado, en cada iteración se va revelando la dirección de correo de *protonmail.com;* el programa entra en una rutina de descifrado donde las dos cadenas son texto en clave. Los creadores consiguen así ocultar la dirección de correo y el resto de los elementos del HTML para mostrarlos solamente cuando se ha producido la infección.

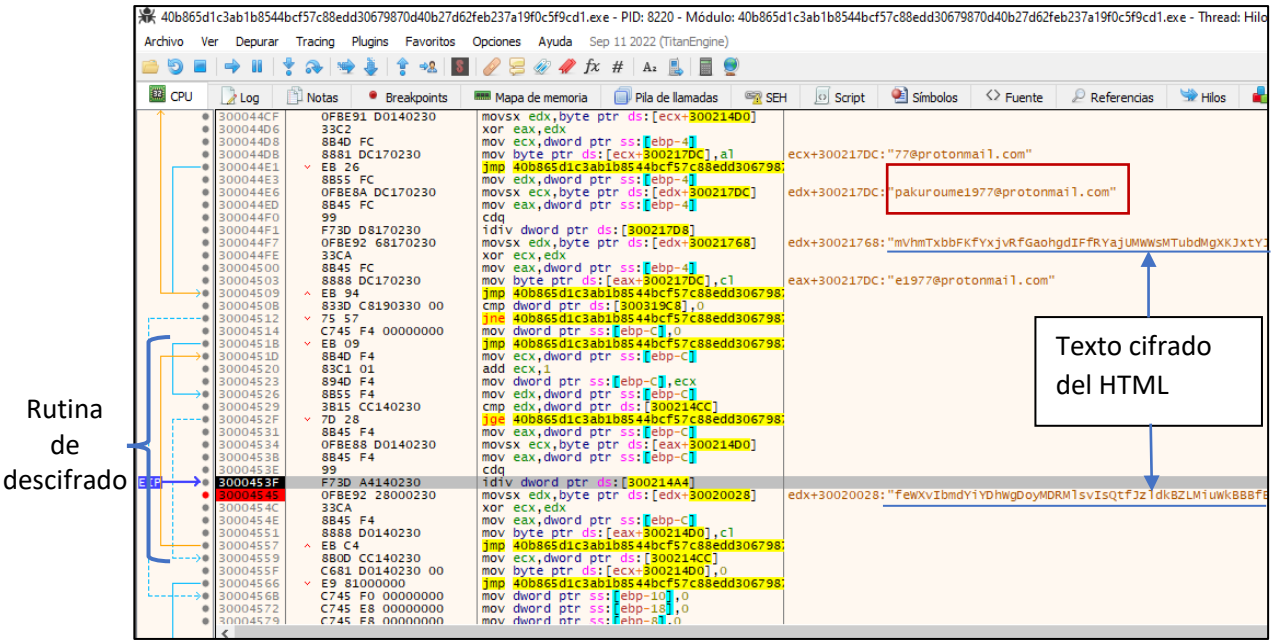


Figura 5.25 Rutina de descifrado que revela el correo electrónico de contacto en tiempo de ejecución.

En último lugar, la tabla 5.3 detalla todas las llamadas a funciones de API que quedaron ocultas a anteriores análisis; son criptográficas y de red. Esta técnica descrita como GetProcAddress spoofing, tiene como finalidad eludir posibles controles de seguridad al no encontrarse funciones peligrosas (o, al menos, sospechosas) en los análisis de un PE. (Babkin. A, 2020).

Criptografía	Red
CryptExportKey	wNetEnumResources
CriptImportKey	wNetOpenEnum
CryptDeriveKey	wNetCloseEnum
CryptGenKey	GetIPNetTable
CryptDecrypt	
CryptDestroyKey	

Tabla 5.3 Llamadas a Funciones de API no detectadas en el análisis estático.

5.5.2- Funciones Criptográficas

Cómo utiliza la criptografía Ryuk se explicará en el capítulo: Uso de la criptografía en Ryuk y medidas de mitigación contra el cifrado de datos. Sin embargo, se expondrán algunos detalles sobre la información obtenida en el desensamblado y depuración.

La clave pública RSA embebida en el código, mostrada en el editor hexadecimal del apartado Capacidades criptográficas, se carga en memoria antes de comenzar el cifrado de datos mediante la función *CryptImportKey* . Observando la figura 5.26 podrá comprobarse que ambas coinciden.

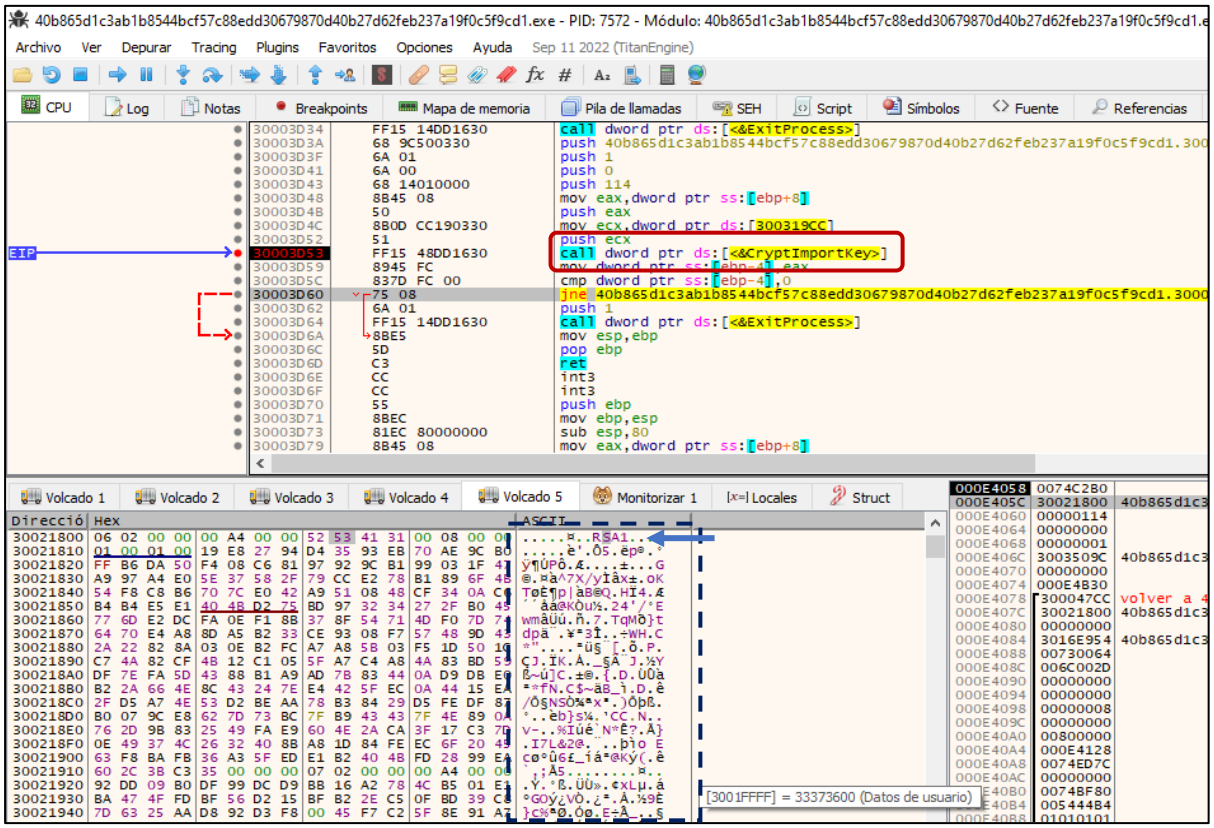


Figura 5.26 Importación de clave pública RSA hacia el proveedor de servicios criptográficos

La clave se transferirá al proveedor de servicios criptográficos y posteriormente solo será accesible a través de un handler. En el proceso de cifrado de datos, los métodos implementados en el código fuente crearán una clave AES aleatoria con CryptGenKey, para,

después, realizar una llamada a *CrypExportKey* y cifrar con la clave pública la mencionada clave AES. Esto puede verse en parte en la figura 5.27.

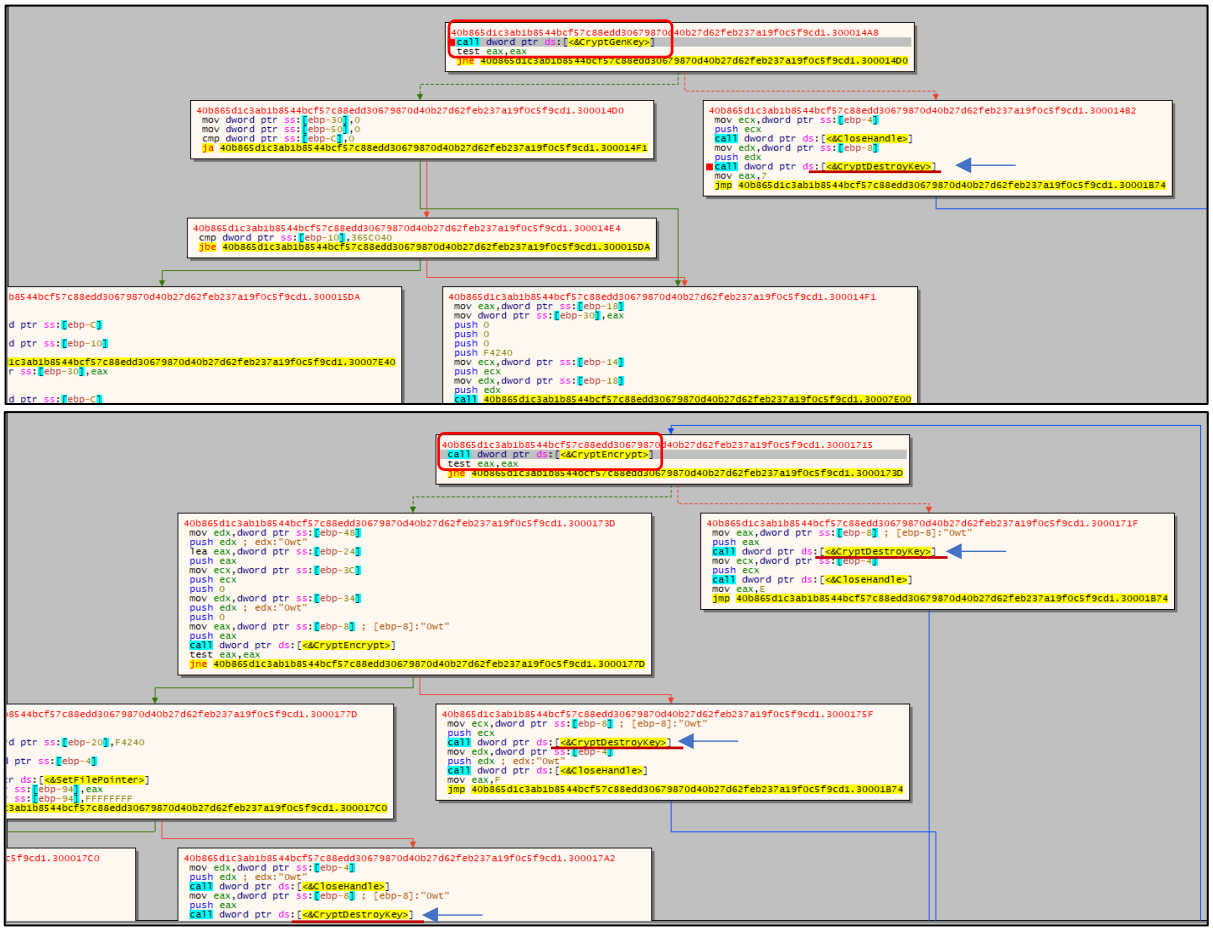


Figura 5.27 Proceso de cifrado de datos, llamadas a CryptGenKey, CryptEncrypt y a CryptDestroyKey en el código fuente

Otra característica importante, también visible en la figura 5.27, son las llamadas a la función *CryptDestroyKey* que los desarrolladores han tenido a bien incluir en todas las instrucciones condicionales. Esto quiere decir que, durante la ejecución del código, en cualquier posibilidad referente al proceso de encriptado se contempla el borrado del handler y la clave criptográfica AES de la memoria del sistema, por lo que, en teoría, un análisis forense de la memoria RAM no sería viable para poder recuperar las contraseñas.

5.5.3- Funciones de red

Tal y como se había observado en el apartado de detección de otros dispositivos, es la instancia o copia ejecutada con el parámetro 8 LAN la encargada de las funcionalidades correspondientes a las redes locales, movimiento lateral. Si se ejecuta el ransomware analizado con dicho parámetro, ejecuta las llamadas a API necesarias para detectar otros nodos en línea.

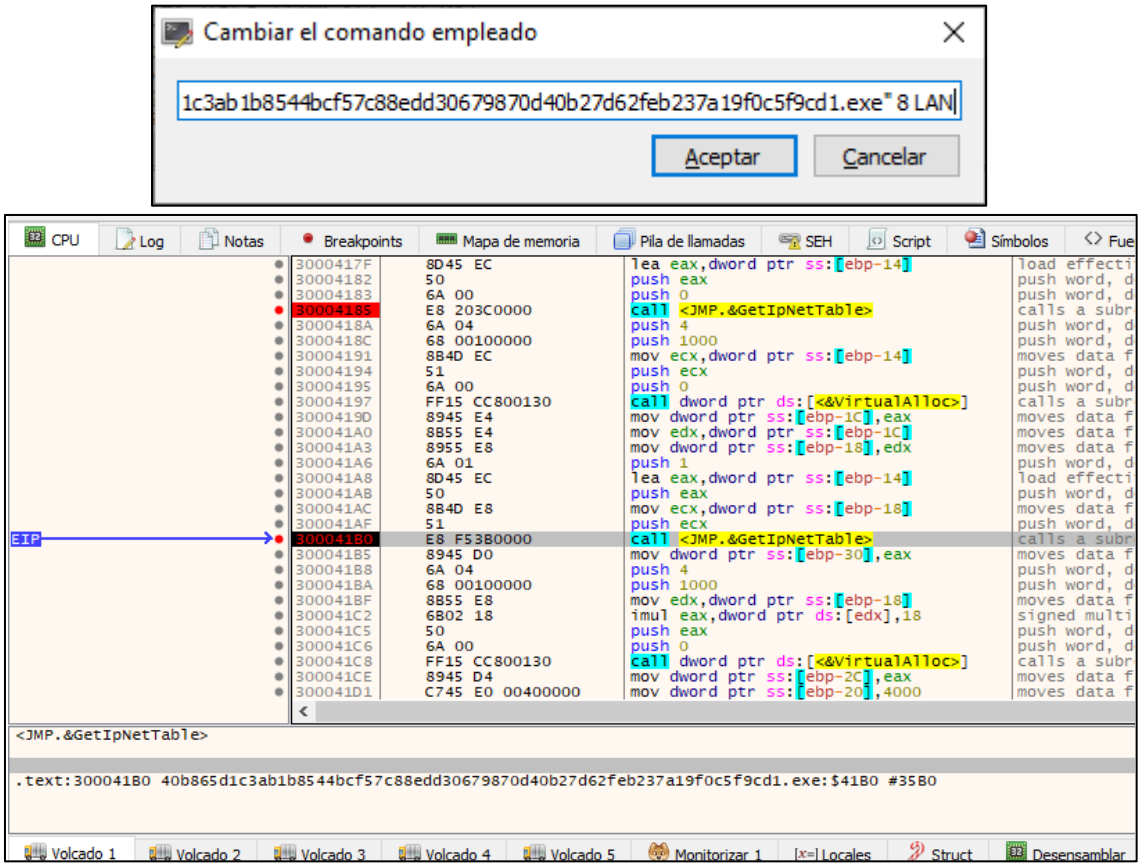


Figura 5.28 Ejecución con parámetro 8 LAN y llamada a GetIpNetTable.

En la figura 5.28 se observa que, con el parámetro 8 LAN se inicia el ransomware y, casi inmediatamente, llama a *GetIpNetTable* incluido en la API *iphlpapi*. *GetIpNetTable* recupera la información de la tabla ARP almacenada en la memoria RAM de la máquina infectada, de este modo, se hace con un listado de direcciones al que enviar las solicitudes Echo. La instancia de Ryuk original iniciada sin el parámetro 8 LAN, no realiza ninguna de estas funciones.

5.5.4- Interacciones con el sistema operativo

En los primeros pasos de su ejecución, antes de recuperar las direcciones MAC de la tabla ARP, Ryuk recopila información sobre el idioma, el nombre del usuario con la sesión iniciada, su identificador de sesión y el nombre del dominio invocando a *winlogon.exe*. Aunque sí consulta el idioma del sistema, no llega a realizar ninguna acción adicional según el lenguaje activo en ese momento. Inmediatamente después sigue una serie de pasos posiblemente destinados a detectar si su código se está ejecutando en un entorno virtualizado. De forma iterativa, a través de *GetProcessHeap* recorre todos los procesos abiertos del sistema según se muestra en la figura 5.29.

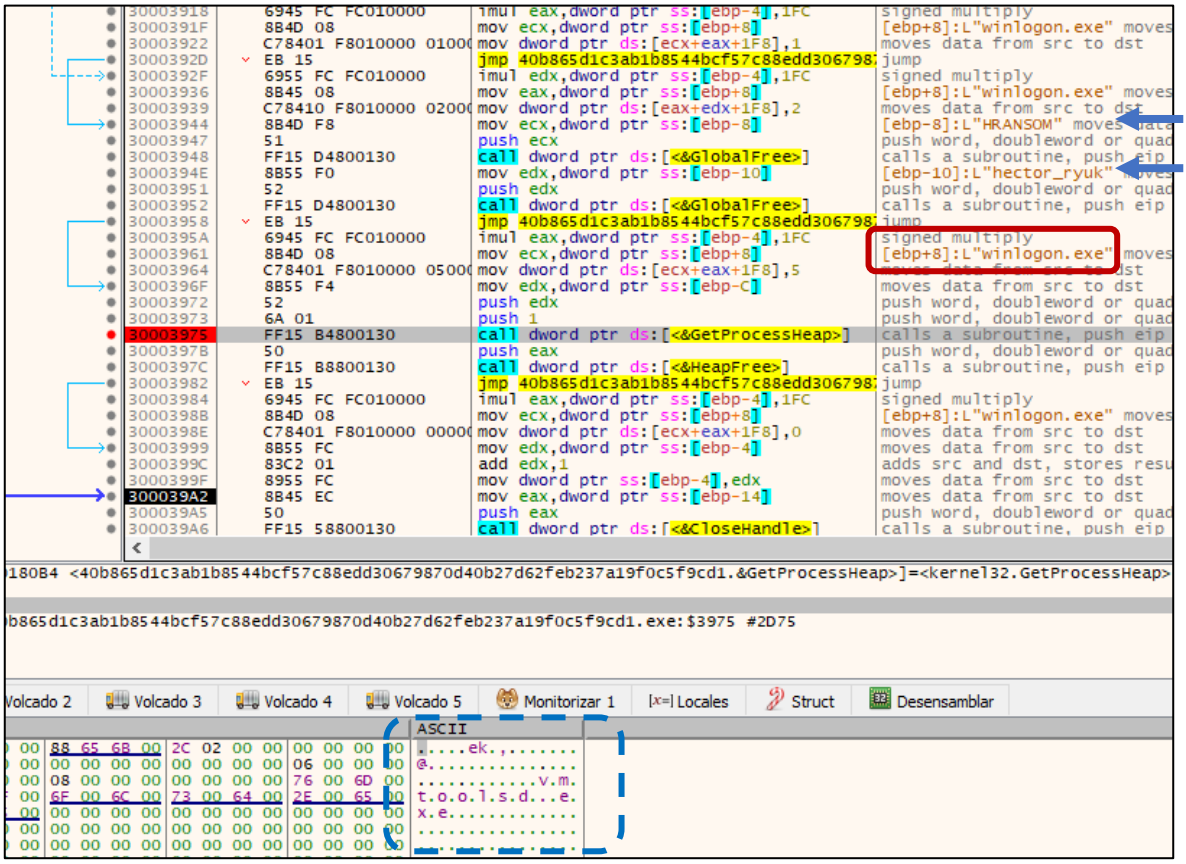


Figura 5.29 Recopilación de datos del usuario e iteraciones entre todos los procesos abiertos del sistema.

Por otro lado, en la muestra analizada, en ningún momento se observó que los nombres de los procesos detectados en el apartado Evasión de software antivirus y procesos se llegasen a

cargar en memoria para una comparación de estos con los que se encontrasen abiertos. Respecto a otros recursos que nunca llegan a usarse, las operaciones de lectura/escritura en el registro de Windows no se llevan a cabo desde las funciones de API *regOpenKey* o *regQueryValue*, por ejemplo. En su lugar se emplean comandos lanzados desde cmd.exe si el binario dispone de privilegios suficientes, como bien indican las observaciones del análisis dinámico y la captura de pantalla de la figura 5.30.

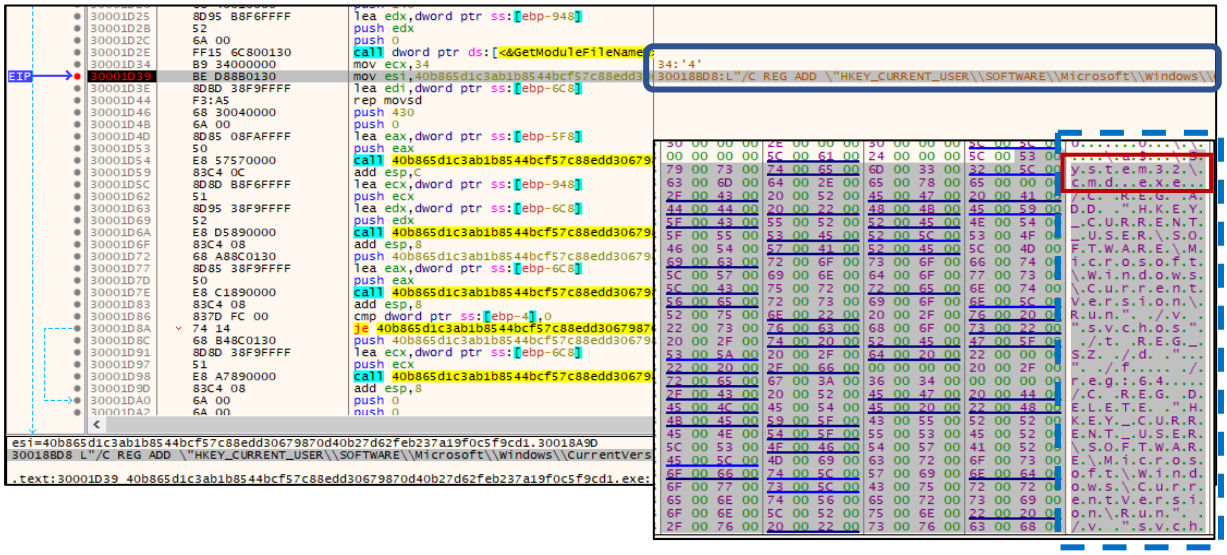


Figura 5.30 Escritura de claves en el registro a través de cmd.exe con un comando incluido como string en el código.

Otros análisis sobre Ryuk disponibles (Centro Criptológico Nacional, 2021) determinan que su espécimen examinado no incluye métodos para asegurar su ejecución una vez reiniciado el sistema operativo. Esta forma torpe de implementar la persistencia, incluidas todas las fallas y fragmentos de código sin referenciar, podrían significar cambios rápidos para adaptar el malware a una nueva campaña.

6. Uso de la criptografía en Ryuk y medidas de mitigación contra el cifrado de datos

El principal objetivo de cualquier ransomware es hacer los datos del usuario y, si es posible, de cualquier otro dentro de la misma red inaccesibles. Las técnicas de encriptación varían entre los diferentes tipos y familias, pero, como se expondrá en este apartado, Ryuk es concienzudo y eficaz para limitar al máximo cualquier posibilidad de revertir el cifrado. Un ejemplo de ello son las constantes llamadas a *CryptDestroyKey* para destruir de la memoria las múltiples claves generadas.

Como posible método para mitigar el cifrado de datos, también se mostrará aquí el desarrollo de una herramienta en forma de aplicación de escritorio que, en algunos casos, puede capturar las llamadas a las APIS criptográficas de Windows y recuperar las claves generadas en implementaciones poco seguras. Para los supuestos en los que no sea posible dicha captura, bloqueará el presunto proceso malicioso y generará un informe con datos del binario.

6.1- Algoritmos criptográficos empleados por Ryuk

Esquemáticamente, la secuencia de acciones que Ryuk sigue para volver inaccesibles los ficheros más sensibles del sistema operativo es la representada por la siguiente figura 6.1

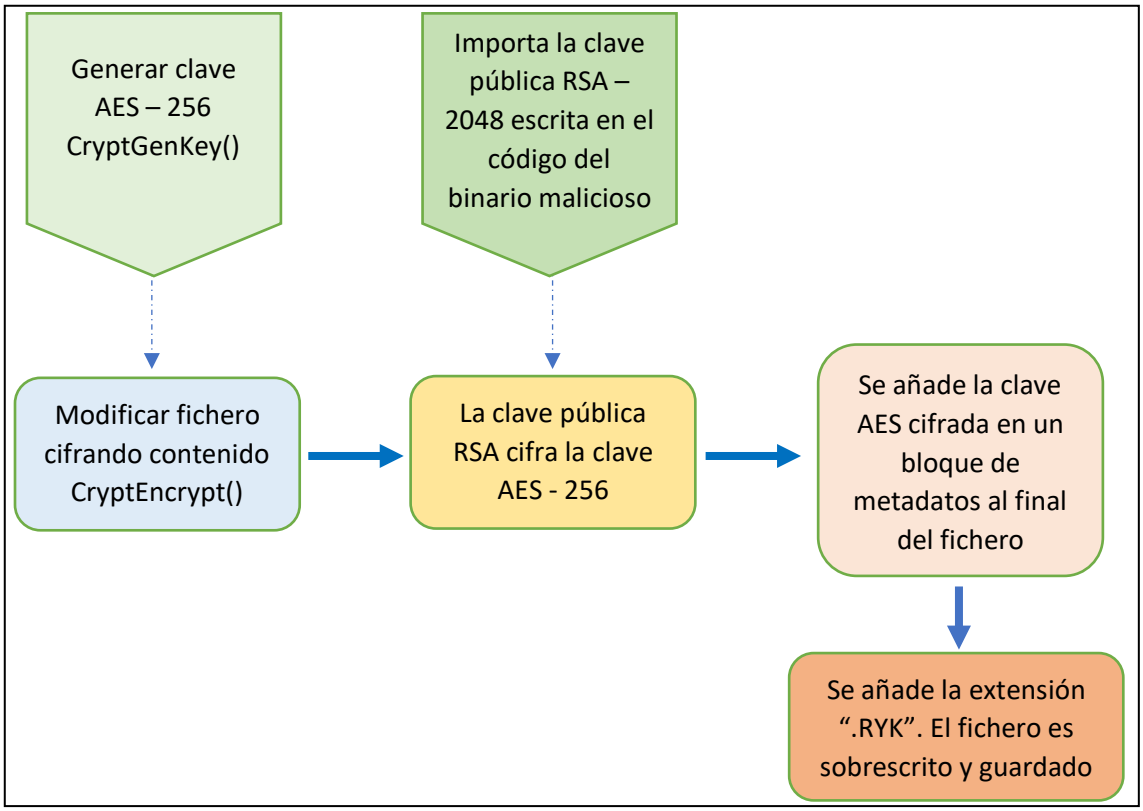


Figura 6.1 Esquema simplificado del proceso de encriptado de ficheros. (Elaboración propia).

A la pregunta de por qué se han escogido estos algoritmos y este procedimiento, se puede contestar de forma sencilla. La cantidad máxima de bytes que puede cifrar RSA viene determinada por el tamaño de la clave. La longitud del criptograma resultante es igual a la longitud del módulo en bytes :

Una clave pública **e** de longitud **n** bits bastaría para codificar $m^e \ (Mod\ N)$ si $2^{n-1} < N < 2^n$

Una clave de 2048 bits podría cifrar 256 bytes, sin embargo, no se debe olvidar que el estándar criptográfico *PKCS#1-v1_5* establece 11 bytes para los parámetros de codificación que se deben restar del tamaño inicial:

$$LongitudMensaje <= \ n - 11$$

La limitación implicaría que cuando el texto a cifrar (en este caso los bits de los ficheros del disco duro) es más grande, debiera dividirse en segmentos de tamaño adecuado para después concatenarlos en el archivo resultante. Obviamente, esta operación es computacionalmente costosa y poco práctica cuando se pueden estar procesando varios gigabytes de información.

Por lo tanto, la solución es utilizar un cifrado de bloque seguro como AES-256 cuyo tamaño de entrada por bloque es de 16 bytes, pero que permite implementar un flujo de datos con el que generar secuencialmente tantos bloques como sean necesarios para abarcar el tamaño total del fichero o mensaje a encriptar. Además, requiere menos memoria y recursos para un proceso que, como ya se ha comprobado, satura la CPU y RAM de una computadora promedio.

La clave RSA privada es propiedad de los atacantes, cuyo modus operandi encaja en un modelo que siguen otros ciberdelincuentes basados en campañas, con compilaciones específicas de su malware donde han generado un par de claves para un objetivo en concreto. (Hassan, N. 2019). Consiguen de esta forma que el contenido sea prácticamente indescifrable evitando en cierto modo que un afectado pague el rescate y publique las credenciales adquiridas.

Respecto a la clave AES-256, no se conocen vulnerabilidades suficientemente relevantes como para considerar revertir el cifrado de Ryuk. Por ejemplo, un ataque de fuerza bruta debería tener en cuenta estos factores:

- El número de Bytes por segundo que una computadora podría ser capaz de descifrar. Esta unidad de medida se corresponde realmente con el Mebibyte (MiB) cuyos múltiplos son potencias de dos y no de diez.

- AES es un algoritmo de cifrado por bloques y el tamaño del bloque es de 2^4 (16 Bytes).

- La complejidad asociada a este problema es exponencial $O\ (m^n)$ y , en el peor de los casos, corresponde con $O\ (2^{256})$. Sin embargo, estadísticamente en una distribución habitual se descubrirá la clave al haber probado la mitad de las posibles combinaciones, por lo que podría establecerse a la mitad del anterior valor:

$$\frac{O\ (2^{256})}{2} \rightarrow\ O\ (2^{255}).$$

Sabiendo esto, una estación de trabajo de gama alta con un procesador con un procesador *AMD Ryzen Threadripper Pro 3990WX*, constaría de una capacidad de cómputo, por ejemplo, en operaciones de encriptado de datos, de 123.801 MIB/Sec (PassMark Software, s.f) que, aunque muy probablemente sea capaz de ofrecer valores superiores si se trata de probar combinaciones aleatorias, puede servir como referencia.

- En bytes por segundo:

$$123.801\ MIB/\sec =\ 129814757376\ bytes/\ sec$$

- Redondeando en la potencia más cercana de dos:

$$129814757376\ \cong 2^{37}\ bytes/sec$$

- Puesto que el tamaño de bloque AES es $16\ bytes =\ 2^4$, para calcular cuántos bloques por segundo podría cifrar y cuántas claves podría probar se dividen las dos potencias:

$$2^{37} : 2^4 \rightarrow 2^{37-4} = 2^{33}$$

- En un mes ininterrumpido de trabajo, que tiene 2678400 segundos, podría comprobar en 31 días:

$2^{33} * 2678400\ =\ 23.007.280.811.212.800 \rightarrow$ 23.007,28 billones de claves por segundo

- Finalmente, el número de meses necesario para llevar a cabo un ataque por fuerza bruta resulta en una cantidad desproporcionada:

$$\frac{2^{255}}{23.007{,}28} = 2.516422739989419 * 10^{60}$$

- Expresando el número completo el resultado es impracticable teniendo en cuenta que cada archivo es cifrado con una clave AES-256 distinta:

2.516.422.739.989.419.153.741.947.518.817.779.811.754.769.696.795.596.431.556.608

6.2- Estructura de los archivos cifrados generados

Las capturas de pantalla de la figura 6.2 constatan como deliberadamente se colocaron algunos archivos para que el ransomware los cifrara. Puesto que recorre los diferentes directorios en orden alfabético, una carpeta cuyo nombre sea, por ejemplo: “aaa”, será la primera en ser procesada. Independientemente del tipo de archivo comprometido, todos poseen un bloque de metadatos al final del fichero que, para la versión de Ryuk analizada comienza con la palabra “HERMES”, lo cual supone una evidencia sobre la relación entre estas dos familias de ransomware.

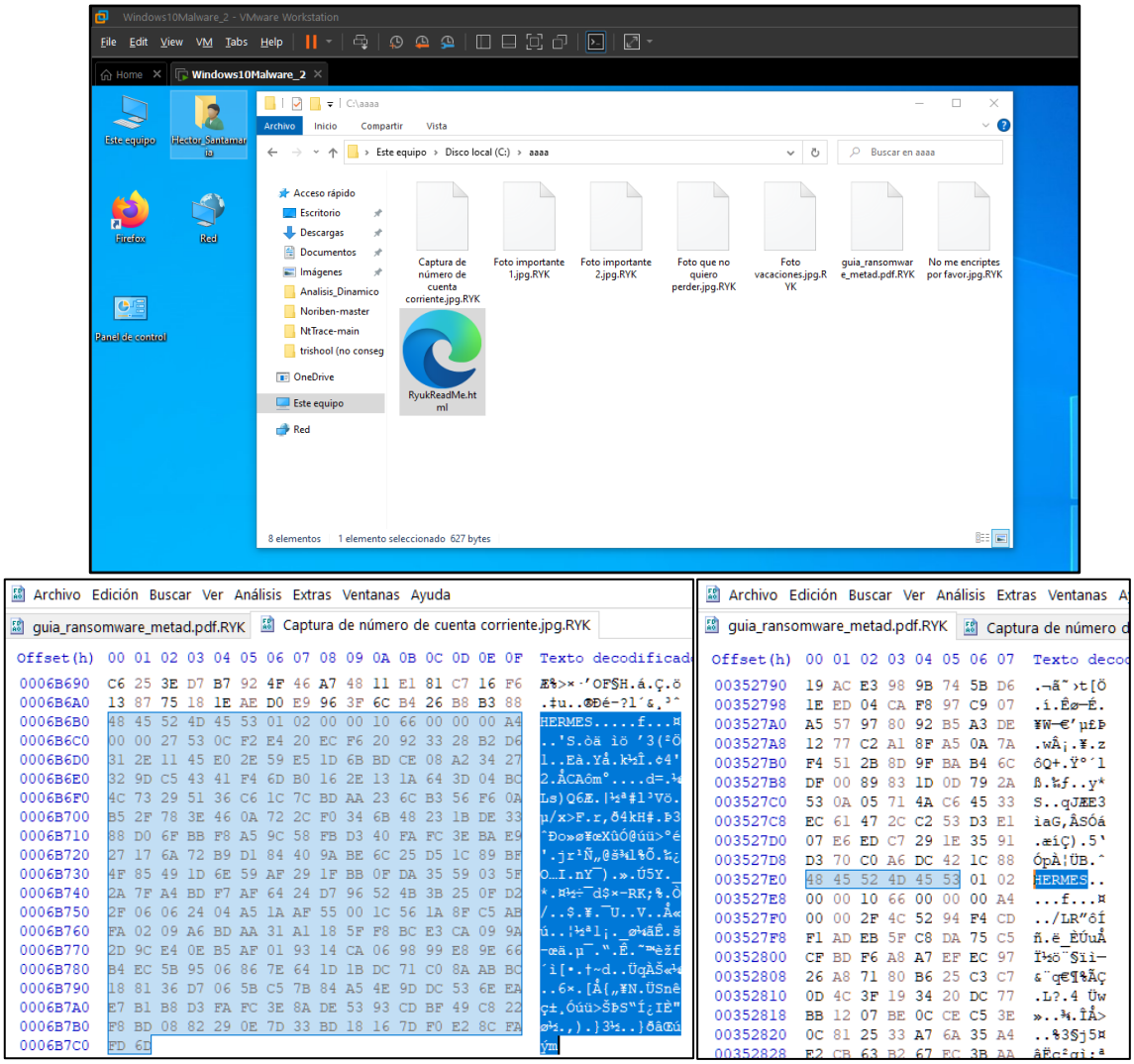


Figura 6.2 Ficheros encriptados por Ryuk y muestra de la sección de metadatos en un editor hexadecimal.

Analizando más detenidamente esta última sección que el ransomware escribe al terminar de codificar el archivo, se pueden observar los siguientes detalles en la figura 6.3 fácilmente comprensibles siguiendo la documentación oficial disponible en la web.

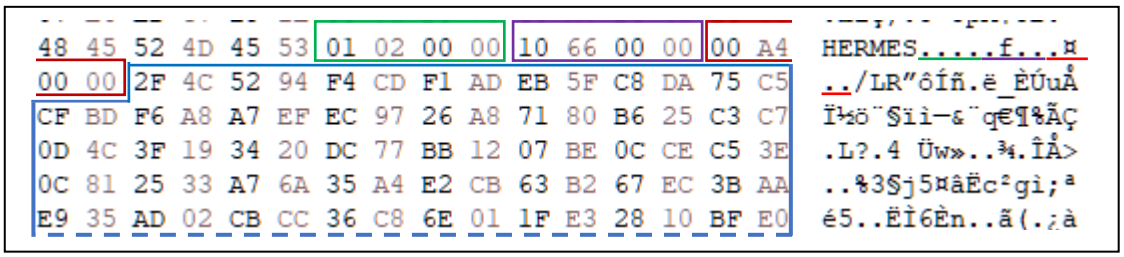


Figura 6.3 Cabecera incluida por Microsoft a la clave AES encriptada que delata detalles sobre la misma.

En la tabla 6.1 se desgranan cada uno de los bloques que componen esa última sección del fichero con la clave criptográfica de cifrado.

Cabecera Blob de clave simple		
Tipo SIMPLEBLOB	Código que comienza por los 4 siguientes bytes que en este caso se utiliza para transferir las claves de sesión desde el CSP (*Cryptographic Service Provider)* al espacio de datos de aplicaciones.	0X01 – 0X02 – 0X00 – 0X00
Valor sessionKeyAlgorithm	Entero de 4 bytes que indica el tipo algoritmo de la clave. Pertenece al tipo SIMPLEBLOB. • El 0x00006610 pertenece a AES-256.	0X10 – 0X66 – 0X00 – 0X00 (se encuentra en formato Little endian)
Valor CALG_RSA_KEYX	Código que describe cómo es la clave pública con la que se ha cifrado la clave simétrica. El 0x 0000A400 pertenece hace referencia al uso de un algoritmo de intercambio de claves públicas RSA de CryptoApi 1.0 o 2.0.	0X00 – 0x00 – 0XA4 – 0X00 (se encuentra en formato Little endian)

Tabla 6.1 Sintaxis de cabecera Blob de clave simple.

Es sencillo averiguar el tipo de clave RSA simplemente seleccionando los caracteres posteriores a la cabecera BLOB. Hacen un total de 256 bytes, lo cual son 2048 bits que

corresponde con el tamaño del estándar RSA-2048 bits. La figura 6.4 indica cuáles son los caracteres que forman la clave AES cifrada, así como su tamaño en bytes.

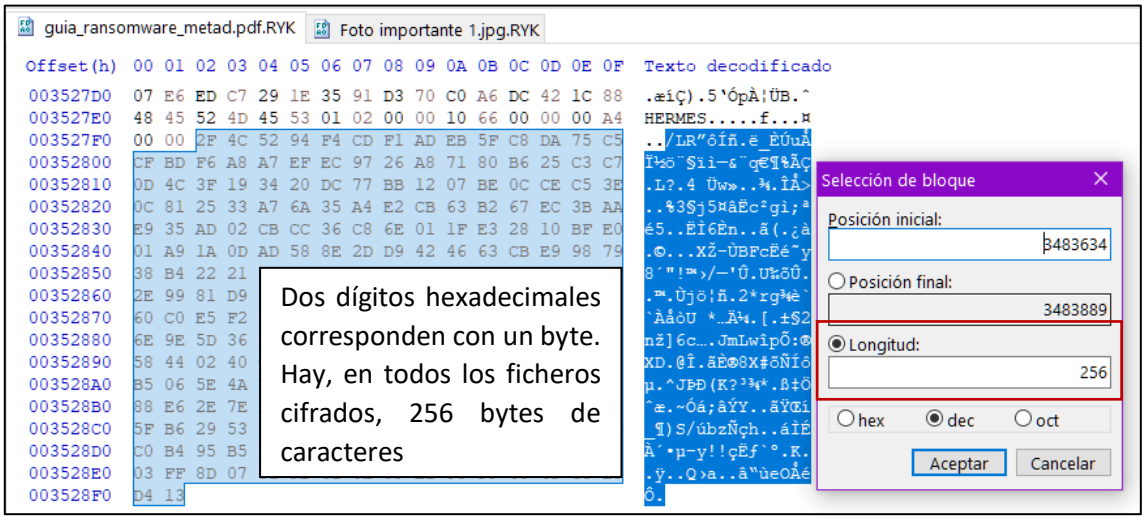


Figura 6.4 El número de bits del estándar RSA utilizado queda patente en código hexadecimal de cualquier archivo cifrado.

Al generar una clave AES para cada archivo se evita cualquier análisis sobre el contenido encriptado que pueda revelar información o que en un análisis forense de la memoria volátil del sistema dicha clave se recupere. De igual modo, las claves AES-256 nunca podrán ser descifradas debido al cifrado asimétrico RSA-2048. Exceptuando considerar examinar un volcado de memoria RAM, la mejor solución puede ser monitorizar los procesos del sistema en busca de llamadas a la CryptoAPI y monitorizarlos.

6.3- Mitigación del cifrado de archivos

A raíz del examinar el comportamiento de este y otros ransomwares, una de las cuestiones que surgieron es si era posible recuperar de alguna forma las claves AES generadas para cada fichero antes de que fuesen cifradas con RSA y utilizarlas posteriormente para recuperar los datos afectados. Una forma podría ser teniendo acceso a los bytes del proceso malicioso, por ejemplo, en un volcado de memoria para un posterior análisis forense, pero, dejando a un lado la dificultad y tiempo que supone, es poco probable una garantía del 100% de recuperación.

Como opción interesante para actuar contra el proceso de encriptado de datos, se diseñó un programa que puede engancharse a las llamadas del malware a una librería del sistema concreta (principalmente las que proveen funciones criptográficas) y extraer el contenido de los parámetros. Técnica comúnmente conocida como “API Hooking”.

Aunque existe una relativamente completa documentación sobre las APIs y funciones implementadas en Microsoft Windows, el API Hooking puede resultar complejo y difícil efectuar, sin embargo, una librería útil que permite hacerlo a alto nivel en varios lenguajes de programación es Deviare2. La herramienta creada aprovechando las librerías Deviare2 está escrita en C# y se llama HectorCryptoHooking. Es realmente muy simple y, aunque ofrece pocas funcionalidades, resulta válida para los supuestos que aquí se van a exponer.

La interfaz gráfica de la herramienta aparece en la figura 6.5 y sus características se enumeran a continuación:

- Permite introducir el nombre del proceso a monitorizar (siempre que este se encuentre activo) para comenzar su monitorización.

- Permite activar una monitorización automática donde detectará los procesos que realicen llamadas a las APIS y métodos para los que se ha programado.

- Genera en un cuadro de texto información sobre los procesos que están generando claves criptográficas mediante funciones de API determinadas, así como los que se han bloqueado automáticamente por realizar esas acciones.

- Escribe en un log con las claves recuperadas en implementaciones para generarlas de forma “poco segura” como *CryptGenRandom*

- Bloquea los procesos que empleen implementaciones seguras a la hora de generar claves, AES o de otro tipo, como *CryptGenKey*. Posteriormente, escribe un log con información detallada sobre los mismos.

- La monitorización se hará únicamente entre una las dos funciones de advapi32.dll (*CryptGenRandom* o *CryptGenKey)* objeto de estudio, para evitar una sobrecarga que genere inestabilidad en el sistema operativo.

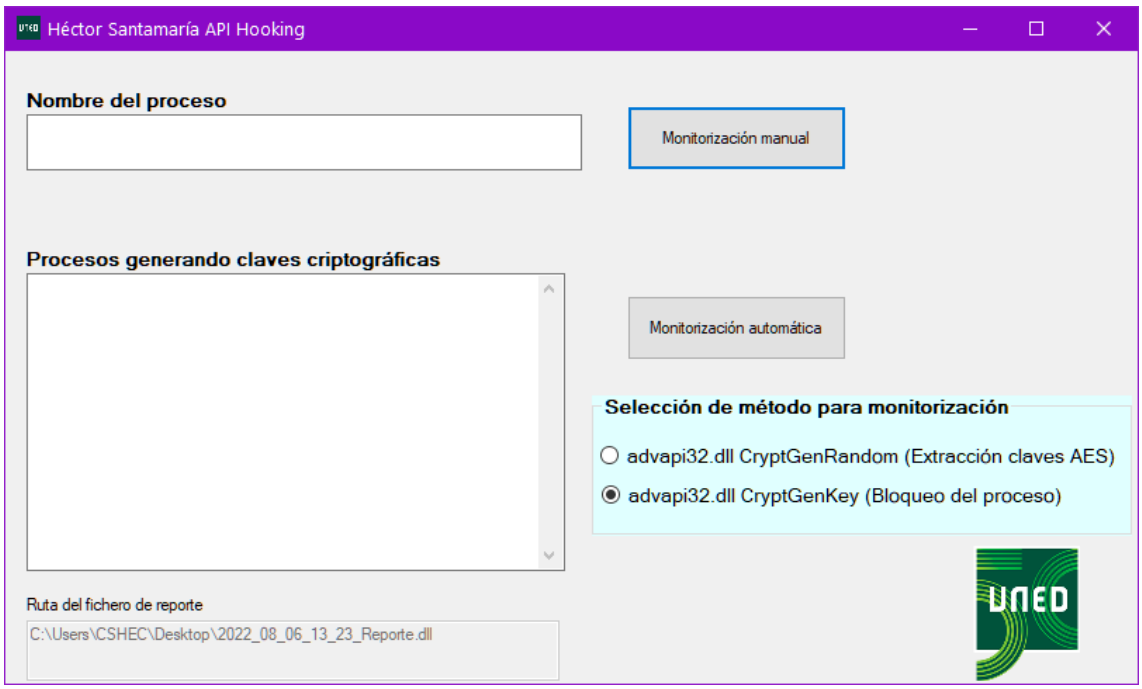


Figura 6.5 Interfaz gráfica de la aplicación desarrollada en C#: HectorCryptoHooking.

Un diagrama explicativo de las funciones de HectorCrytoHooking se encuentra en la figura 6.6. El programa monitoriza dos tipos de llamadas a funciones de la API de Windows, en función de cada una llevará a cabo la extracción de la contraseña del buffer de memoria o el bloqueo del proceso extrayendo información relevante. Independientemente de a qué función se enganche siempre se escribirá un log en el escritorio del sistema afectado con los datos recuperados.

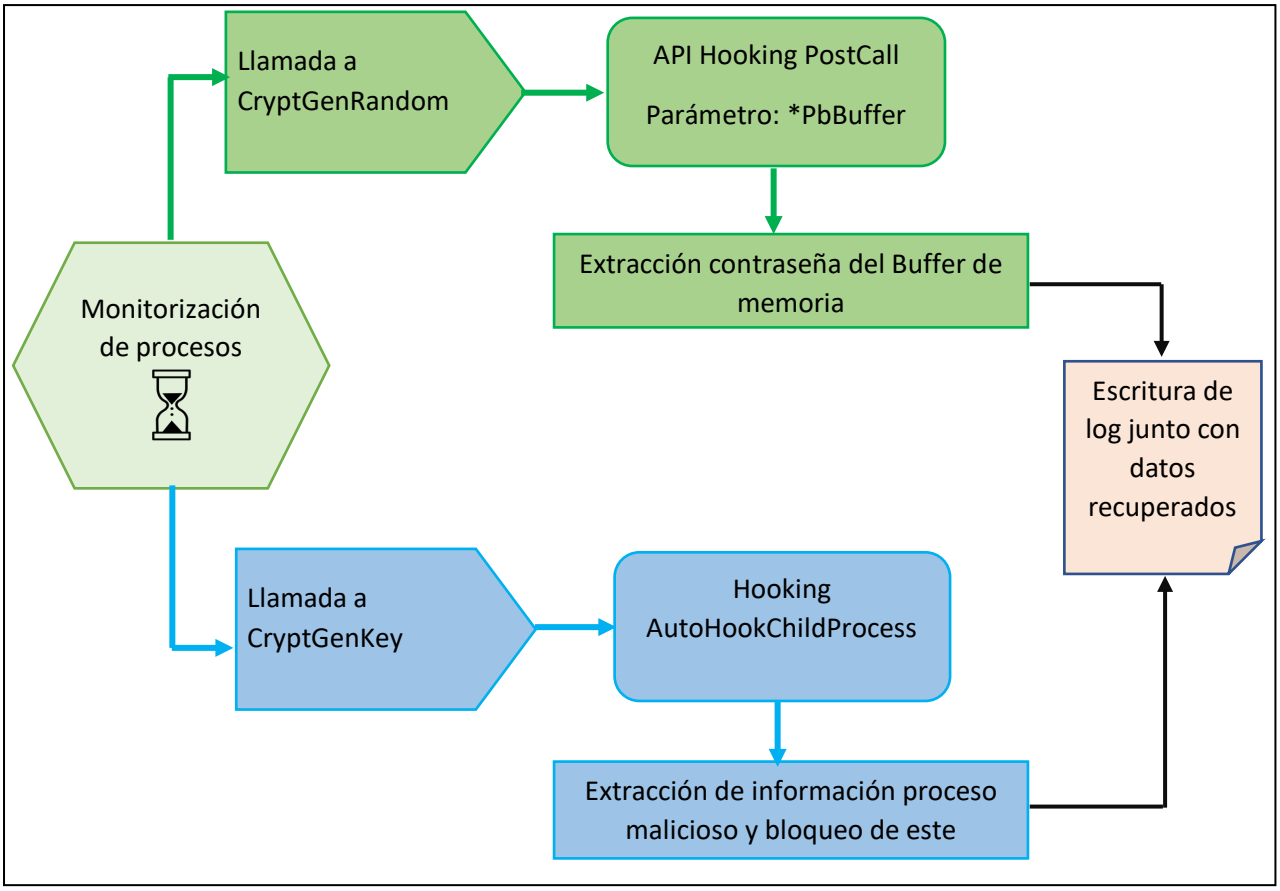


Figura 6.6 Diagrama explicativo sobre el funcionamiento de HectorCryptoHooking. (Elaboración propia).

6.3.1- Partes del código más relevantes

Se describe aquí de forma somera algunas de las partes más relevantes del código de la aplicación, útiles para entender cómo realiza el API Hooking y la interacción con las APIs de Windows soportadas en la herramienta.

6.3.1.1- Form_Load() Carga inicial de la aplicación

En la carga del formulario, sobre el cual se muestra un fragmento del código en la figura 6.7, se instancian e inicializan variables con diversas funciones:

- Creación de la ruta del fichero de reporte siempre en el escritorio junto con una marca de tiempo.

- *FileStream()* y *StreamWritter()* Escribirán líneas de texto en el fichero de reporte.

- Un *ArrayList()* para almacenar los procesos que realicen llamadas a las API seleccionadas junto con un *Timer* en un hilo propio para actualizar el *TextBox* con el listado de procesos de la interfaz gráfica.

- Un objeto *NktSpyMgr()* como componente fundamental para "espiar" los procesos del sistema operativo sobre los que después se efectuará el enganche.

```
NktSpyMgr spyMgr = new NktSpyMgr();
spyMgr.Initialize();
```

Figura 6.7 Fragmento de código correspondiente a la carga inicial de HectorCryptoHooking.

6.3.1.2- Búsqueda manual y Búsqueda automática

La búsqueda manual recibe un proceso como parámetro y lo busca entre un listado que devuelve el *SpyMgr()*, que ya se encuentra examinando todos los que existan. Si el proceso existe llama directamente al método *cargarMonitorizacion(NktProcess proceso).*

```
String nombreProceso = textBoxNombreProceso.Text;
.......
NktProcessesEnum listaProcesos = spyMgr.Processes();
NktProcess procesoLista = listaProcesos.First();
        while (procesoLista != null)
        {
            if (procesoLista.Name.Equals(nombreProceso,....)
            {
                 cargarMonitorizacion(procesoLista);
            ....
             procesoLista = listaProcesos.Next();
            ....
```

```
spyMgr.OnProcessStarted += OnProcessStarted;
spyMgr.OnProcessTerminated += OnProcessTerminated;
```

Figura 6.8 Fragmento de código correspondiente a la búsqueda manual y automática de HectorCryptoHooking.

La búsqueda automática implementa dos interfaces que hacen la función de *listener* para detectar todos los procesos que comienzan y que terminan. Cuando un proceso comience, llamará al método *cargarMonitorizacion(NktProcess proceso)*. La figura 6.8 contiene código, a modo de ejemplo, que implementa las funciones descritas.

6.3.1.3- CargarMonitorizacion (NktProcess proceso)

El comienzo de la monitorización tiene lugar en este método donde se crea realmente el enganche con la librería y la función de la API de Windows seleccionada. Puede observarse en la figura 6.9 que hay dos enganches instanciados referenciando las dos funciones creadoras de claves criptográficas que son objeto de estudio: CryptGenKey y CryptGenRandom.

```
NktHook enganche_CryptGenKey =
spyMgr.CreateHook("advapi32.dll!CryptGenKey",(int)(eNktHookFlags.flgOnly
PostCall));

NktHook enganche_CryptGenRandom =
spyMgr.CreateHook("advapi32.dll!CryptGenRandom",(int)(eNktHookFlags.flgA
utoHookChildProcess));


  ____________________________________________________________________


enganche_CryptGenKey.OnFunctionCalled += OnFunctionGenKeyCalled;
enganche_CryptGenRandom.OnFunctionCalled += OnFunctionGenRandomCalled;


  ____________________________________________________________________


enganche_CryptGenKey.Attach(proceso, true);
```

Figura 6.9 Fragmento de código correspondiente a la monitorización de un proceso concreto en HectorCryptoHooking.

Estos objetos se asocian a una interfaz distinta con tres parámetros, uno de esos parámetros proporciona toda la información de la llamada. La última fase es enlazar los objetos NktHook con el proceso que se monitoriza.

Es importante destacar que puede añadirse un flag: *flgOnlyPostCall*, *que aparece también en la figura 6.9,* para especificar cómo se producirá la captura. En este caso, la captura de los parámetros de la llamada a función en el *Hooking* es posterior a dicha llamada, por lo que así se pueden obtener las variables de salida. En caso de querer tener la certeza de extraer los parámetros de entrada, se podría utilizar *flgOnlyPreCall***.** Existen otros muchos otros flags para realizar hooking a los procesos hijo que el padre pueda generar, solo en llamadas asíncronas y para versiones de API 32 o 64 bits.

6.3.2- Extracción de claves en ransomware con implementaciones poco seguras (CryptGenRandom)

Dentro de los mecanismos que proporciona Microsoft Windows para crear claves criptográficas aleatorias, se encuentra *CryptGenRandom()*, función incluida en *Wincrypt.h* dentro de la API *Advapi32.lib*. Consta de los parámetros añadidos a la tabla 6.2:

CryptGenRandom()		
Entrada (HCRYPTPROV)	hProv	Handle hacia el CSP
Entrada (DWORD)	dwLen	La cantidad de bytes de los datos generados.
Entrada - Salida (BYTE)	*pbBuffer	Un puntero al buffer con los bytes que conforman la clave criptográfica.

Tabla 6.2 Sintaxis de función CryptGenRandom() para C++.

Para esta implementación, aunque uno de los parámetros de entrada sea un *handle* hacia el proveedor de servicios criptográficos de Windows, simplemente se indica la longitud en bytes que debe poseer la clave (*dwLen*) y ésta es almacenada en un buffer que no está dentro de ningún espacio de memoria protegido en el CSP, resultando fácilmente recuperable al examinar la memoria del proceso que realiza la llamada.

6.3.2.1- Desarrollo realizado

En HectorCryptoHooking se desarrolló el método *OnFunctionGenRandomCalled()* desde el cual su código inspecciona el contenido de la memoria a la que apunta **pbBuffer*. Primero se deben recuperar los parámetros de *CryptGenRandom()* mediante la variable que en la llamada al método Deviare2 pasa como parámetro: `INktHookCallInfo` ***`callInfo`***: Después vuelca la memoria del proceso desde el SpyManager de Deviare2 y se instancia un objeto de tipo Byte[] reservando un tamaño equivalente a la longitud de la clave.

```
INktParamsEnum pms = callInfo.Params();
uint longitudClave = pms.GetAt(1).ULongVal; //dwlen
INktParam paramClave = pms.GetAt(2);//pbBuffer

_____________________________________________________________________

INktProcessMemory procMem = spyMgr.ProcessMemoryFromPID(proc.Id);
var datosBuffer = new byte[longitudClave];

_____________________________________________________________________

//Creamos un puntero para enlazar datosBuffer con la lectura de la
memoria del proceso remoto enganchado (Hooking)
GCHandle handleBuffer=GCHandle.Alloc(datosBuffer,GCHandleType.Pinned);
IntPtr punteroBuffer = handleBuffer.AddrOfPinnedObject();
procMem.ReadMem(punteroBuffer, paramClave.PointerVal, longitudClave);

_____________________________________________________________________

stringClave = Convert.ToBase64String(datosBuffer);
```

Figura 6.10 Fragmento de código correspondiente a la extracción de claves en CryptGenRandom() con HectorCryptoHooking.

Posteriormente, se reserva memoria local (perteneciente a HectorCryptoHooking) y se llena *datosBuffer* con los bytes de la clave, leyendo la memoria recuperada del proceso remoto enganchado (que está utilizando la función *CryptGenRandom()*) para acabar convirtiendo los bytes a una codificación Base64. El código perteneciente a estas fases está resumido en la figura 6.10.

Toda la lógica anteriormente representada se materializa en un sistema efectivo para extraer en Base64 posibles claves AES de 256bits. Comprobar que este sistema funciona es sencillo con un script para PowerShell (*Generador de clave con CryptGenRandom.ps1*) que invoque *CryptGenRandom()* para crear una contraseña. El resultado puede verse en la figura 6.11 y comprobar que, efectivamente, lo recuperado por el programa y lo generado por el script coinciden.

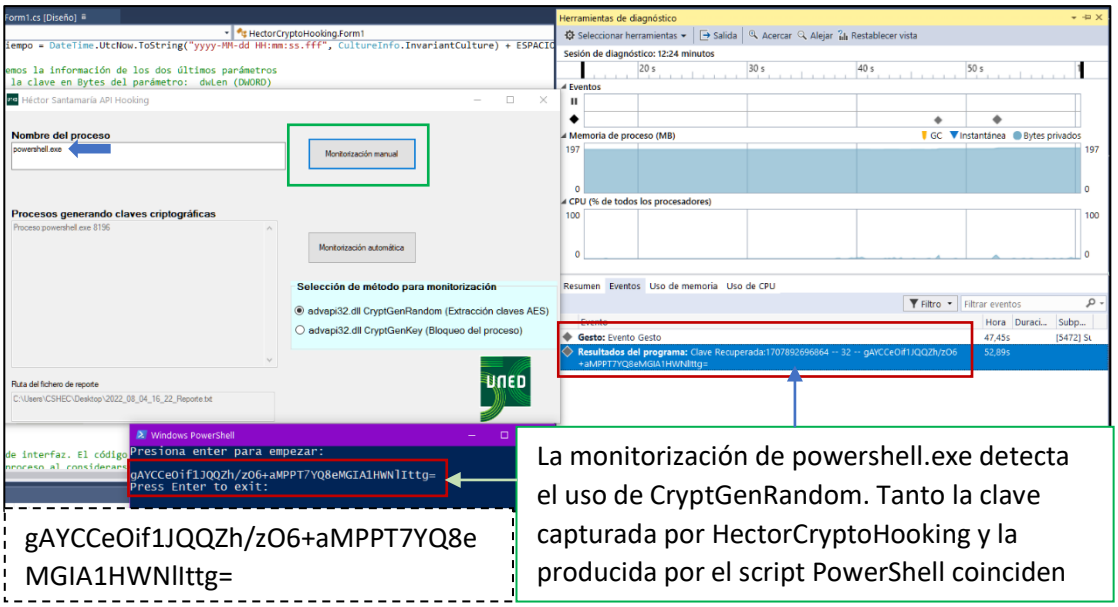


Figura 6.11 Captura de clave AES mientras se depura la aplicación.

6.3.2.2- Pruebas realizadas con ransomware Lilith

Un ransomware de nueva creación en el momento de la redacción de este trabajo de fin de máster, es Lilith. Con él se demostrará como la herramienta puede extraer las contraseñas para cualquier ransomware con que utilice este sistema. Su aparición se sitúa en torno a julio de 2022, emplea *CryptGenRandom()* para crear una clave y su propia rutina de encriptado. Excluye, entre otros, los directorios Windows y Archivos del programa, así como archivos con extensión *.Dll* (Cyble Inc, julio de 2022) .

Teniendo dicha información, se descarga una muestra con estas características:

Nombre de archivo
f3caa040efb298878b99f883a898f76d92554e07a8958e90ff70e7ff3cfabdf5.exe

Tamaño de archivo
240,640 bytes

Registrado por primera vez
2022-07-07 07:17:47 UTC

MD5
b7a182db3ba75e737f75bda1bc76331a

SHA-1
cf0fe28214ad4106c48ec5867327319eaa82b3c3

SHA-256
f3caa040efb298878b99f883a898f76d92554e07a8958e90ff70e7ff3cfabdf5

Vhash
025086655d55551515555088z6b!z

Authentihash
af1277c0a9317ee818c9fa702be46282d1ce361f79fb90a762ea48a938d318bb

Imphash
261a21398ec064ef3b57c9095d03d0e8

Rich PE header hash
8d5098bc7847076def68b9bb7449f7a0

SSDEEP
3072:UFnHQbj1nLJXNpVi+1Wqv/Sk3vfpHpLbVS7+Yf1VBUW9WEMNny2NQ+IMI:WHQbj1nVJgq/bHU/fvBUWSC

TLSH
T143346C5AFAA249EAE5B6C134CC65480699F13C140720DB7F87201A21DF27BF1DE7AB25

La figura 6.12 captura el instante en el que Lilith muestra una consola de comandos vacía durante su ejecución siendo el cifrado de los ficheros muy rápido, quizá por no utilizar algoritmos de cifrado complejos. HectorCryptoHooking y los archivos necesarios para su ejecución se almacenan en ProgramFiles para evitar ser alterados y el fichero de reporte se renombra con la extensión *Dll* por el mismo motivo. El programa detecta el uso de la API específica y extrae todas las contraseñas generadas sin demasiada dificultad.

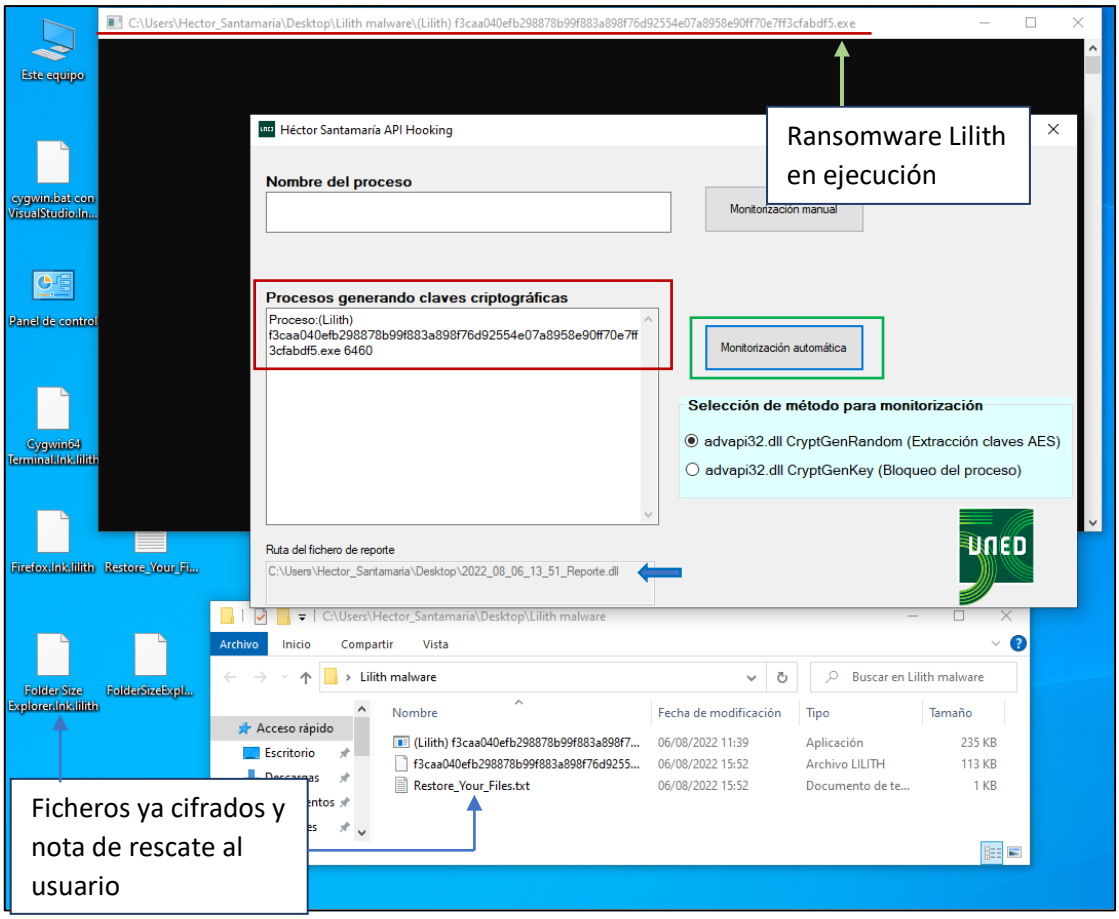


Figura 6.12 Ransomware Lilith detectado y captura de claves en funcionamiento.

En menos de un minuto *2022_08_06_13_51_Reporte.dll*, expuesto en la figura 6.13, tiene almacenadas 4.400 contraseñas en formato Base64, además el reporte añade cuál es su longitud: 32 bytes (256 bits). El primer valor numérico se trata únicamente del handle al CSP de Windows. Para cada ejecución del ransomware será diferente.

A pesar de que no se ha planteado el desarrollo de la herramienta para también incluir el documento que cifra, sí queda registrada una marca de tiempo que se podría cotejar con cada fecha de modificación de los ficheros y proporcionar alguna estimación sobre qué clave corresponde a cada uno.

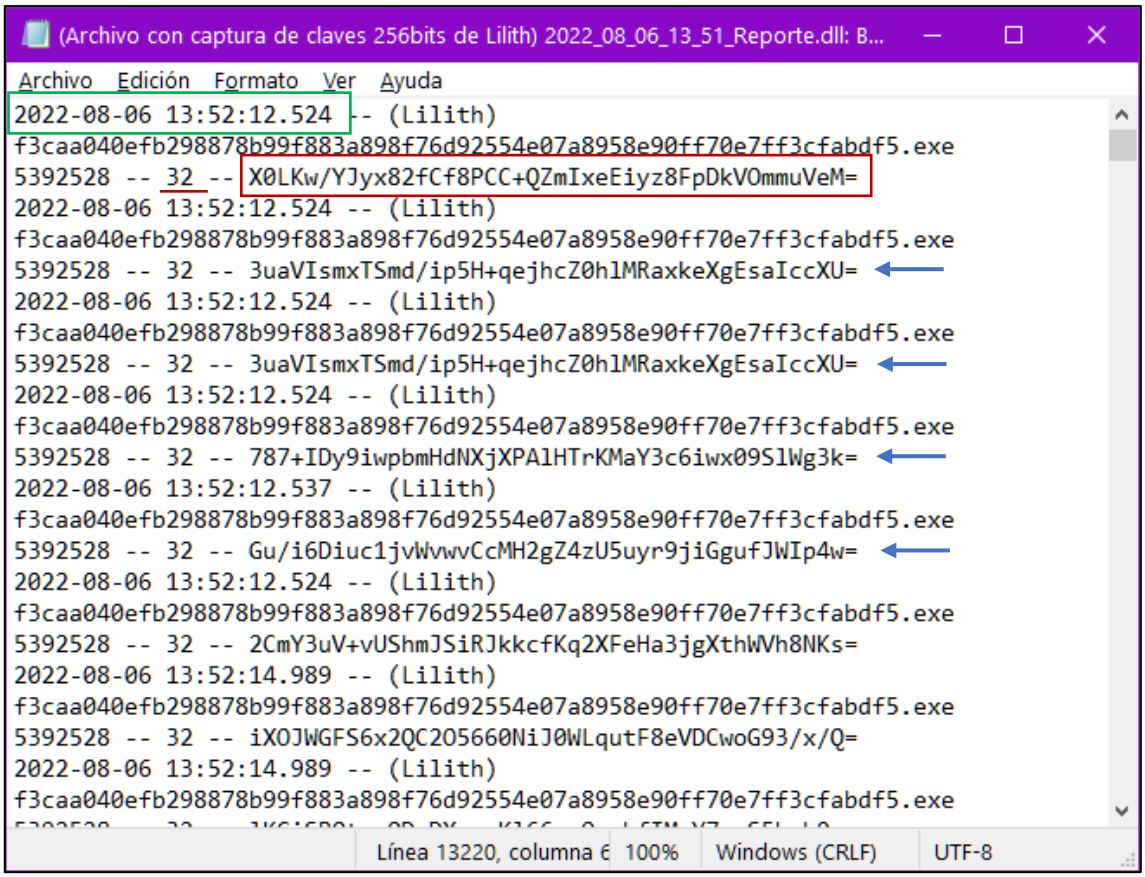


Figura 6.13 Fichero con contraseñas extraídas de 256 bits junto con marca de tiempo.

6.3.3- Extracción de claves en ransomware con implementaciones seguras (CrypGenKey)

Otro de los sistemas que proporciona Windows para producir claves criptográficas aleatorias, es *CryptGenKey(),* también correspondiente a la librería *Advapi32.lib* y a la cabecera *Wincrypt.h* La función consta de los parámetros descritos en la tabla 6.3:

CryptGenKey()		
Entrada (HCRYPTPROV)	hProv	Handle hacia el CSP
Entrada (ALG_ID)	Algid	Valor numérico que identifica el algoritmo empleado.
Entrada (DWORD)	dwFlags	Tipo de clave generada
Salida (HCRYPTKEY)	*phKey	Dirección de memoria hacia el handle de la nueva clave

Tabla 6.3 Sintaxis de la función CryptGenKey para C++.

En este caso, los valores relevantes pueden ser *AlgId* y el contenido del puntero *phKey* que debería contener la clave generada. Sin embargo, *phKey* representa un tipo de datos HCRYPTKEY definido en la API *Wincrypt* (encabezado *wincrytp.h*). Lo que cualquier programa puede obtener es un identificador o handle de la clave criptográfica generada que se encuentra dentro del espacio restringido del Proveedor de Servicios Criptográficos (CSP). Este sistema seguro imposibilita la recuperación o acceso a las claves almacenadas.

En algunas pruebas realizadas, donde se llevó a cabo un procedimiento similar al apartado de extracción en implementaciones poco seguras, estos son algunos de los datos conseguidos:

```
022-07-20 21:47:10.010

(ALG_ID) 26128 --   (typename)HCRYPTKEY   (value)7230176 --
wFCyc5Cts3PAhrJzcMezc8CYsnNATLJzkGKyc4CYs3M=

2022-07-20 21:47:11.744

(ALG_ID) 26128 --   (typename)HCRYPTKEY   (value)7229728 --
wFCyc5Cts3PAhrJzcMezc8CYsnNATLJzkGKyc4CYs3M=

2022-07-20 21:47:10.012

(ALG_ID) 26128 --   (typename)HCRYPTKEY   (value)7230944 --
wFCyc5Cts3PAhrJzcMezc8CYsnNATLJzkGKyc4CYs3M=
```

El valor de AlgId se encuentra representado como decimal. La conversión en hexadecimal lo transforma a 0x00006610 y corresponde con el identificador CALG_AES_256 que puede tomar un tipo de datos ALG_ID.

Respecto al puntero de tipo HCRYPTKEY, simplemente muestra el identificador del handler a la clave que provee el CSP. Si se extrae el contenido que posee el espacio de memoria del puntero **phKey*, el resultado en BASE64 es el mismo para cada archivo cifrado, por lo que no aporta ninguna información significativa ni corresponde con clave alguna.

6.3.3.1- Desarrollo realizado

Sabiendo que con los medios y conocimientos de que dispone el alumno es imposible recuperar las claves AES, se optó por desarrollar la rutina mostrada en la figura 6.14. Su propósito es extraer la mayor cantidad de información posible y cerrar el proceso

automáticamente. El código se encuentra en el método *OnFunctionGenKeyCalled()*, encargado de extraer características del objeto *INktProcess* que recibe como parámetro, así como algunos datos de la cabecera y secciones:

```
"Nombre: "+ proc.Name
"PID: " + proc.Id
"UserName: + proc.UserName
"Ruta: " + proc.Path

.....

NktStructPEFileHeader fileHeader = proc.FileHeader();

"Características:" + fileHeader.Characteristics
"Número de Secciones:" + fileHeader.NumberOfSections
"Tamaño de la cabecera opcional:" + fileHeader.SizeOfOptionalHeader

.....

NktStructPESections secciones = proc.Sections();

for(int i = 0; i<secciones.Count; i++){
"Sección: " + secciones.Name[i]
 ____________________________________________________________________

stringClave = Convert.ToBase64String(datosBuffer);

 ____________________________________________________________________

if (proc.IsActive)
    proc.Terminate();
```

Figura 6.14 Fragmento de código correspondiente a la extracción de información en CryptGenKey() y posterior finalización del proceso con HectorCryptoHooking

Como último paso Deviare2 permite finalizar el proceso, siendo en ese momento cuando se notifica sobre el proceso bloqueado y se vuelcan los datos recogidos al reporte. En caso de existir una infección por el ransomware, se evita que el fichero sea cifrado por Ryuk o Lilith cambiando la extensión *.txt* por *.dll*

Las pruebas realizadas con Ryuk verifican que, efectivamente, el programa desarrollado detiene el proceso malicioso y también el proceso hijo e hilos generados a partir de la copia que el ransomware hace de sí mismo casi al mismo instante de su ejecución. Esto último es

facilitado por el flag *flgAutoHookChildProcess* que se añade cuando se instancia el objeto NktHook para preparar el enganche.

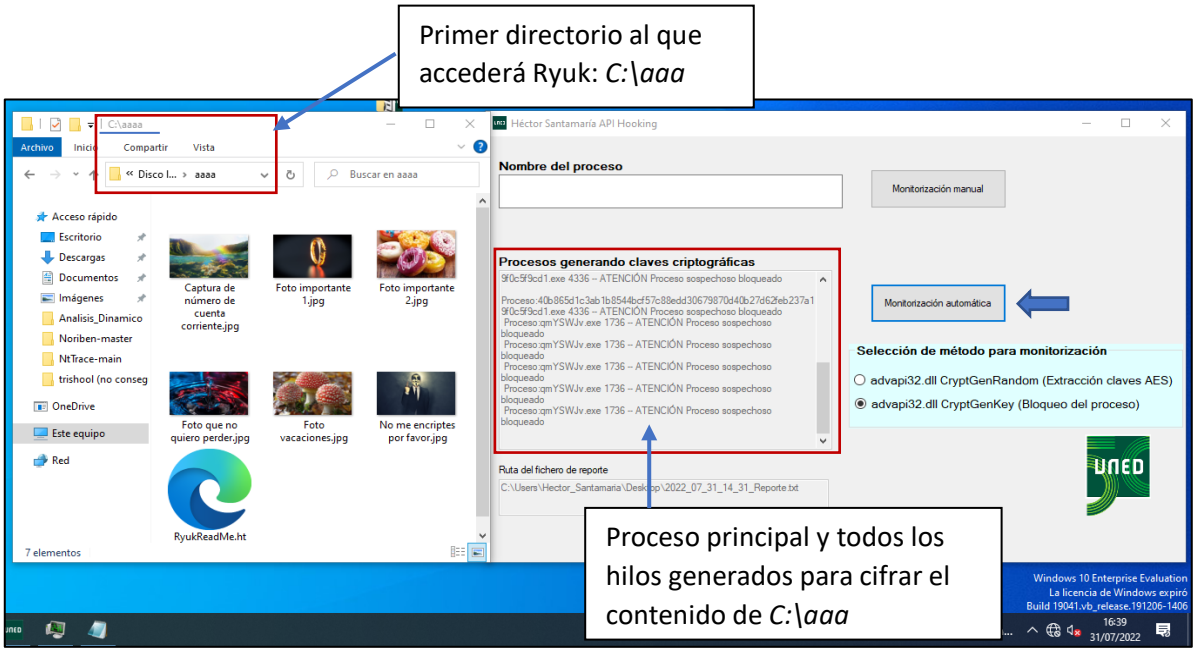


Figura 6.15 Muestra de cómo HectorCryptoHooking ha detenido los procesos que utilizan CryptGenKey de advapi32.dll evitando así el cifrado de archivos por Ryuk.

La figura 6.15 representa gráficamente cómo Ryuk consiguió escribir su habitual archivo *RyukReadMe.html* en la carpeta preparada para que fuese su primer acceso. Sin embargo, tanto el proceso principal como el proceso hijo fueron detenidos en el momento de utilizar la API CryptGenKey evitando cualquier cifrado. Cada hilo generado (seis hilos para las seis fotografías) aparece indicado en la aplicación, además del proceso padre.

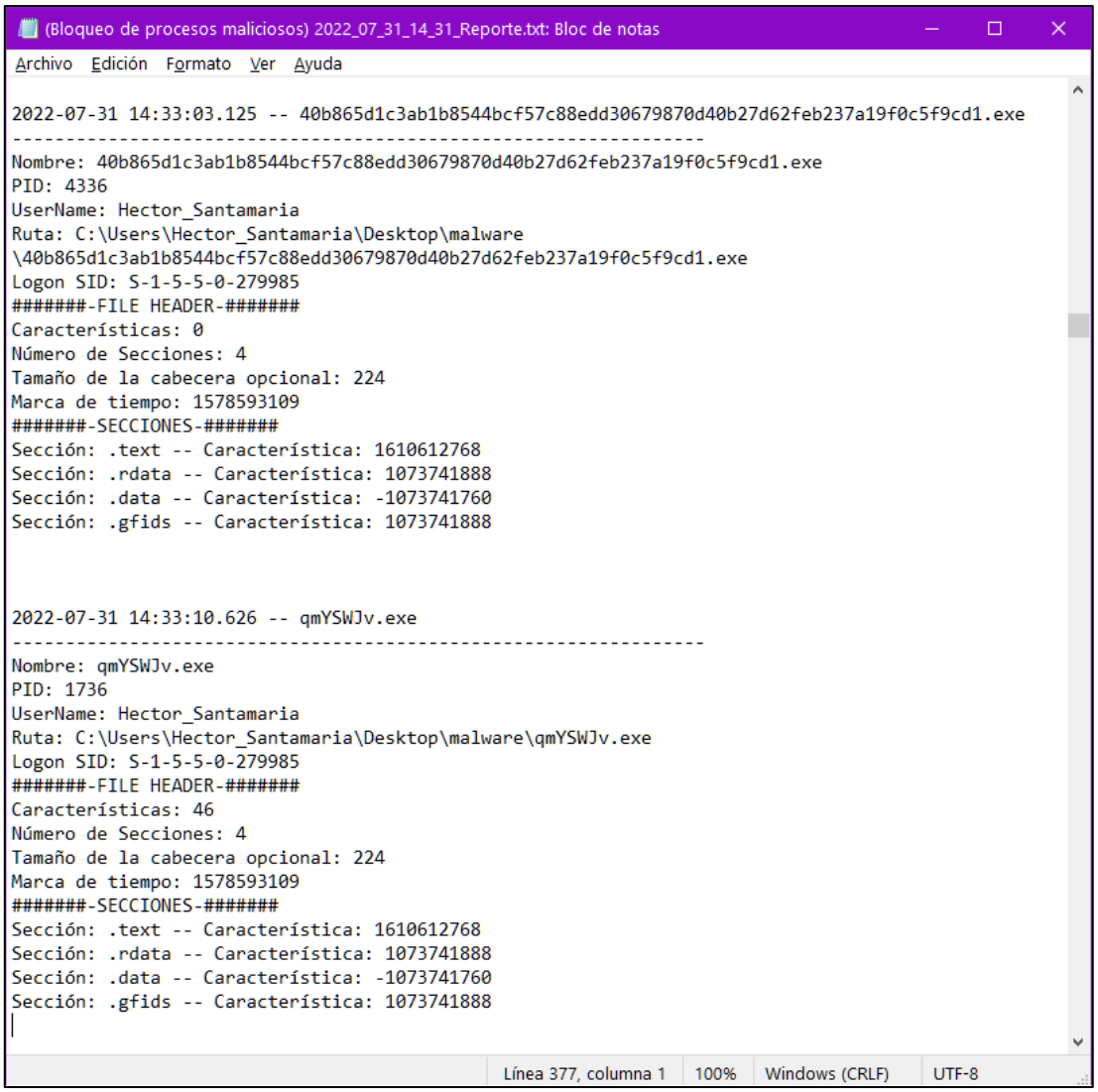


Figura 6.16 Reporte generado por HectorCryptoHooking al bloquear los procesos que utilizan CryptGenKey de advapi32.dll.

El fichero de reporte almacenado en el escritorio contiene la información representada en la figura 6.16. Los datos ya se habían recabado en el análisis estático. Por ejemplo:

- La fecha de compilación del archivo PE en milisegundos. Es el 9 de enero de 2020.
- El tamaño de la cabecera opcional está en bytes y es de 224.
- Los valores de las características de cada una de las secciones están expresados en decimal
    - 1073741888 – 0x040000040
    - 1610612768 – 0x060000020
    - -1073741760 – 0x0C0000040

No obstante, también existen datos de utilidad para un analista, como el nombre y ubicación del proceso, el nombre de usuario que lo inició, el PID, su localización, la fecha en la que ocurrió el evento y el identificador de seguridad SID.

Las APIs que aprovecha este ransomware ya no se recomiendan y están (o deberían estar) en desuso. Actualmente Microsoft proporciona la Cryptography API: Next Generation (CNG) más rápida, segura y sencilla de implementar, por lo que *CryptGenRandom* puede ser sustituida por *BCryptGenRandom* y *CryptGenKey* por *BCryptGenerateKeyPair* que permite generar claves asimétricas y abarcan muchos más algoritmos de cifrado. Si los autores de este malware tuvieron interés por infectar sistemas operativos antiguos como Windows XP o Server 2003, habrían optado por la opción correcta, ya que las nuevas APIs no están disponibles en esas versiones.

7. Uso de técnicas de Machine Learning con características estáticas del ransomware

Existe una tendencia creciente en incorporar técnicas de inteligencia artificial y aprendizaje automático para la detección de malware y ransomware. Por ejemplo, productos de seguridad de detección de amenazas (IDS) y prevención de amenazas (IPS), como puede ser *XGen Security* de Trend Micro, *Intercept X Endpoint* de Sophos, o soluciones antivirus domésticas como *Avast* y *Kaspersky*, incorporan Machine Learning para ofrecer protección contra posibles amenazas. En este apartado se van a tratar de generar dos modelos mediante dos algoritmos de árbol de clasificación distintos, capaces de predecir cuándo un archivo binario es ransomware o no basándose en las características del archivo PE, información en campos de cabeceras, número y tamaño de secciones, etc.

Al haberse realizado el análisis estático, dinámico y de código, antes de abordar la predicción de binarios, se comprende mejor cuáles de esas características podrían ser más útiles que otras.

Los dos elementos fundamentales necesarios son: una librería o framework diseñada para el aprendizaje automático que incluya, al menos, los algoritmos más populares de clasificación (regresión logística, árboles de decisión) y un set de datos adecuado con características de archivos binarios pertenecientes a ransomware y no relacionados con este tipo de malware.

- La librería escogida es Scikit-lean, de código abierto y diseñada para su uso en el lenguaje de programación Python, incluye algoritmos de clasificación interesantes que son adecuados para este cometido. Los algoritmos de clasificación son útiles cuando se trata de dar una respuesta a un problema dentro de un conjunto de posibilidades previamente definidas. Dentro de ese escenario encaja el objetivo de este trabajo para distinguir si se está ante un fichero PE que encriptará sin consentimiento los archivos del sistema informático o se trata de un programa inofensivo.

- El conjunto de datos o *dataset* proviene del trabajo realizado por los investigadores Ashraf, A. y Aziz, A. (2019). En este caso, el tipo de análisis en el que se centrará esta parte

del proyecto es el análisis estático, por lo que se escogerá el primer set de datos mencionado.

Se decidió emplear algoritmos de clasificación supervisado por tres motivos:

- Al ser aprendizaje automático supervisado se conoce previamente la respuesta que nos dará la función y la motivación del proyecto insta a distinguir entre programas dañinos o benignos, por lo que resulta esencial poder asignar los datos de entrada en una de esas dos variables.

- Un algoritmo de clasificación es idóneo para planteamientos con predicciones finitas. En este caso es un simple problema binario.

- Los árboles de decisión u otros algoritmos basados en ellos, destacan por su rapidez y su rendimiento.

Dentro de las posibilidades que deja el aprendizaje automático supervisado basado en árboles de decisión, los algoritmos empleados que mejor resultado ofrecieron son Random Forest y XGBoost, ambos basados en árboles de decisión.

Random Forest

Un algoritmo de clasificación similar al árbol de decisión, pero con la ventaja de combinar múltiples árboles evaluando la salida de cada uno ellos. El resultado con más apariciones será la respuesta que ofrezca al problema.

Este método, consistente en crear gran cantidad de árboles y trabajar conjuntamente, mejora el algoritmo de árbol de decisión tradicional que solo se entrena con una muestra de datos cada vez. También hay que destacar que en cada árbol solamente se evaluará un subconjunto al azar de características y no todas las que se transmitan en el conjunto de entrenamiento.

XGBoost

Este algoritmo se encuentra en una librería de código abierto y sus siglas significan *Extreme Gradient Boosting*. El sistema de funcionamiento está basado en el aprendizaje automático de árbol de decisión, pero su enfoque y mejoras son tales, que se le considera como uno de los mejores algoritmos para problemas de regresión y clasificación por encima del Árbol de Decisión tradicional o Random Forest.

En su caso, genera múltiples modelos predictivos en un bucle que en principio no serán muy eficientes, no obstante, a medida que continúa generando más, estos recogen los resultados de los modelos (árboles realmente) anteriores para mejorar los siguientes, siendo cada vez más precisos. En los primeros momentos de la ejecución hay diferencias significativas entre los árboles "débiles" y los siguientes que van siendo más "fuertes" y, una vez que ya no hay prácticamente diferencia entre ellos, pues no existen posibilidades de mejora, el algoritmo termina.

7.1- Características escogidas

A continuación, se detallarán cuáles son esas características con una breve descripción de las mismas y, en el caso de haber sido seleccionadas fuera al margen de las investigaciones científicas tenidas en cuenta en el estado del arte [1] [2], un comentario con el porqué de su importancia. Por lo tanto, no se utilizan las 90 recopiladas en el conjunto de datos y habrá algunas más relevantes que otras.

La mayoría de las características están incluidas en la cabecera COFF (Common Object File Format). Esta cabecera se encuentra en los archivos ejecutables específicos que maneja Windows y se incluyó a partir de la aparición de Windows NT en 1994. Consta de una sección estándar y secciones opcionales donde también se hallan características seleccionadas.

MajorOperatingSystemVersion

Se trata del número de la versión principal del sistema operativo obligatorio más viejo en el que el programa podría funcionar. Existe el valor MinorOperatingSystemVersion que se trataría del número de versión secundario. Por, ejemplo un valor igual a “6” indica que la versión mínima puede tratarse de Windows Vista, Windows 7 o Windows 8. Teniendo el código referente a MinorOperatingSystemVersion, un “1” ya especifica la codificación completa del sistema operativo: “6.1”, que es exactamente Windows 7.

Ullah et al. (2020) determinó que esta era una de las características más relevantes en su algoritmo de clasificación basado en árboles de decisión para detectar determinados tipos de ransomware. En este caso, solamente se va a utilizar el número de versión principal y no Minor Operating System Version, ya que es posible que un mismo binario pueda ejecutarse, como requisito mínimo, en plataformas Windows similares altamente compatibles;

NumberOfRvaAndSizes

Especifica la cantidad de entradas de tipo DataDirectory y tamaños. Los “Data Directory” pueden ser como máximo 16 y contienen punteros a los datos de las secciones. Cada una de estas entradas RVA indica la localización correspondiente a las direcciones absolutas de la

memoria virtual del proceso. Conociendo el valor NumberOfRvaAndSizes, cuyo tamaño son 8 bytes, es posible saber cuál es la cantidad de estos directorios que maneja el archivo PE, lo cual es relevante si los binarios ransomware comparten una estructura similar.

Por otro lado, las direcciones virtuales que se desvían de los valores típicos son un indicador de que el PE está utilizando una modificación para evadir su detección por parte de software de detección de amenaza o prevención de intrusiones. (Lewis et al., 2020).

MajorImageVersion

Es un valor que indica la mayor versión del fichero de imagen ejecutable y lo define el desarrollador. Aunque en el trabajo de Lewis et al. (2020) no es una característica relevante para la detección de malware con el algoritmo XGBoost, Ullah et al. (2020) argumenta que gran cantidad de programas no maliciosos tienen más variedad de versiones. Esto último cobra sentido al considerar que el software benigno comercial, normalmente, se actualiza con frecuencia y el número de versión es mucho más grande que 0.

MajorLinkerVersion - MinorLinkerVersion

Muestra al sistema operativo las versiones del enlazador, el primero la principal y el segundo la secundaria. El término *Linker* (enlazador) hace referencia a cómo se mapea el archivo binario en la memoria virtual. Solamente Ullah et al. (2020) demuestra que las dos características son relevantes en los algoritmos de clasificación testados para detectar ransomware (que no incluyen XGBoost).

NumberofSections

Almacena el número de secciones ubicadas en la tabla de secciones. Según el criterio del alumno, esa tabla resulta de especial importancia porque contiene especificadas todas las partes del fichero ejecutable que serán cargadas en memoria. Este campo se refiere al número realmente indica el tamaño de la tabla.

Al igual que ocurre con el campo *SizeOfHeaders,* la arquitectura del ransomware puede ser bastante similar, sobre todo entre variantes de una misma familia distando de la de un programa convencional. Si el ransomware no está empaquetado, probablemente muchos de los binarios maliciosos de este tipo no se desvíen de unos valores estándar.

SizeOfStackReserve

Este valor informa del tamaño de la pila que se va a reservar durante la inicialización del programa. Ullah et al. (2020) lo incluye entre los más importantes.

Characteristics

Muestra los atributos del archivo. Este campo tiene unos posibles valores hexadecimales ya definidos. Por ejemplo, el valor 0x0020 significa que el programa puede manejar direcciones de más de 2 GB y el valor 0x1000 indicaría que el binario es un archivo del sistema y no un programa de usuario. El malware puede servirse de este atributo para que el sistema operativo interprete el archivo de manera interesada.

En total hay 17 marcas posibles de las cuales algunas están obsoletas y otra reservada para un uso futuro aún no definido. Como se observó en el análisis estático de la muestra de ransomware, estos detalles en los binarios pueden ser una fuente de información importante, por lo que se incluye en set de datos final.

AddressOfEntryPoint

Indica la dirección de memoria del punto de entrada al cargarse el binario en memoria. Dependiendo del tipo de imagen (un programa, una DLL, …) podría corresponder con la dirección inicial. Su valor será cero si no hay ninguno definido. Se agrega al dataset porque, tal y como se argumentó en la característica NumberOfRvaAndSizes, es factible que a las direcciones de memoria se les aplique un desplazamiento para evitar detección por parte de software antivirus y no corresponda con un valor estándar. (Lewis et al., 2020).

SizeOfHeaders

Representa el tamaño de la cabecera opcional del fichero binario en bytes. Estas serían las necesarias para la correspondiente implementación COFF y su extensión específica para Windows. Su tamaño abarca desde el principio del archivo hasta el final de la tabla de secciones.

Es incluida entre las características a tener en cuenta porque, respecto a un archivo ejecutable no malicioso y otro ransomware, puede haber una notable diferencia entre el tamaño de las cabeceras y tabla de secciones al asemejarse las implementaciones maliciosas en cuanto a número y tamaño de secciones.

ImageBase

Contiene la dirección de memoria en la que se cargará el primer byte del programa ejecutable. El sistema operativo no siempre tiene porqué cargar el programa en esa dirección ya que puede haber otro con el mismo valor ImageBase, en ese caso usará una diferente. En el estudio de Lewis et al. (2020), es una de las características con más peso en la clasificación con XGBoost de malware. Además, La dirección ExportRVA (Export relative virtual address) es seleccionada como relevante en el artículo de Ullah et al. (2020) y esta es relativa con respecto la dirección de ImageBase en cualquier archivo PE.

DllCharacteristics

Encabezado con una serie de posibles valores en hexadecimal que definen bajo qué circunstancias una librería DLL inicializa una función. Por ejemplo, un valor correspondiente a 0x0040 indicaría que el fichero DLL se puede reubicar en tiempo de carga.

Ullah et al. (2020) incluye esta característica en el set de entrenamiento que emplean para detectar ransomware y Lewis et al. (2020) la escoge, pero mostrando que su peso en XGBoost es muy bajo en comparación con el resto.

Subsystem

Almacena el subsistema requerido para ejecutar archivo binario. Existen varios valores del 0 al 16. Por ejemplo, el valor 2 describe que utiliza la interfaz gráfica de usuario (GUI) de Windows. El 16 indica que debe ser ejecutado como aplicación de arranque del subsistema de imágenes.

Es habitual en los programas para Windows normales que presenten una interfaz gráfica, mientras que la ejecución del ransomware puede ser totalmente transparente al usuario y no integran esas funcionalidades. Además, un valor desconocido (IMAGE_SUBSYSTEM_UNKNOWN – 0) o reservado para las funciones del sistema Windows como dispositivos en modo kernel y procesos nativos del sistema (IMAGE_SUBSYSTEM_NATIVE – 1) harían de ese binario algo sospechoso.

Subsystem es incluida únicamente en el estudio de Lewis et al. (2020) para identificar cualquier binario malware, pero, a priori, no representa ninguna ventaja concreta a la hora de clasificar solo ransomware. A pesar de ello se incluye en el data set, ya que podrá complementarse con características que sí son más influyentes para esa tarea

Num_Suspecious_Sections

Los autores del dataset empleado han ideado esta medida cuantificando las secciones distintas a “rsrc”, “data”, “code” y “rdata”. Es considerado importante, ya que si se encontrase una sección como UPX0 o UPX1, sería clasificada como sospechosa y con acierto, además, ya que, en ocasiones, se suele emplear *crypters* y *packers* para ocultar ransomware y otros tipos de malware. Tanto en el set de datos final como en el script para la extracción de características de binarios *Static_Feature_Parser.py*, se genera este dato.

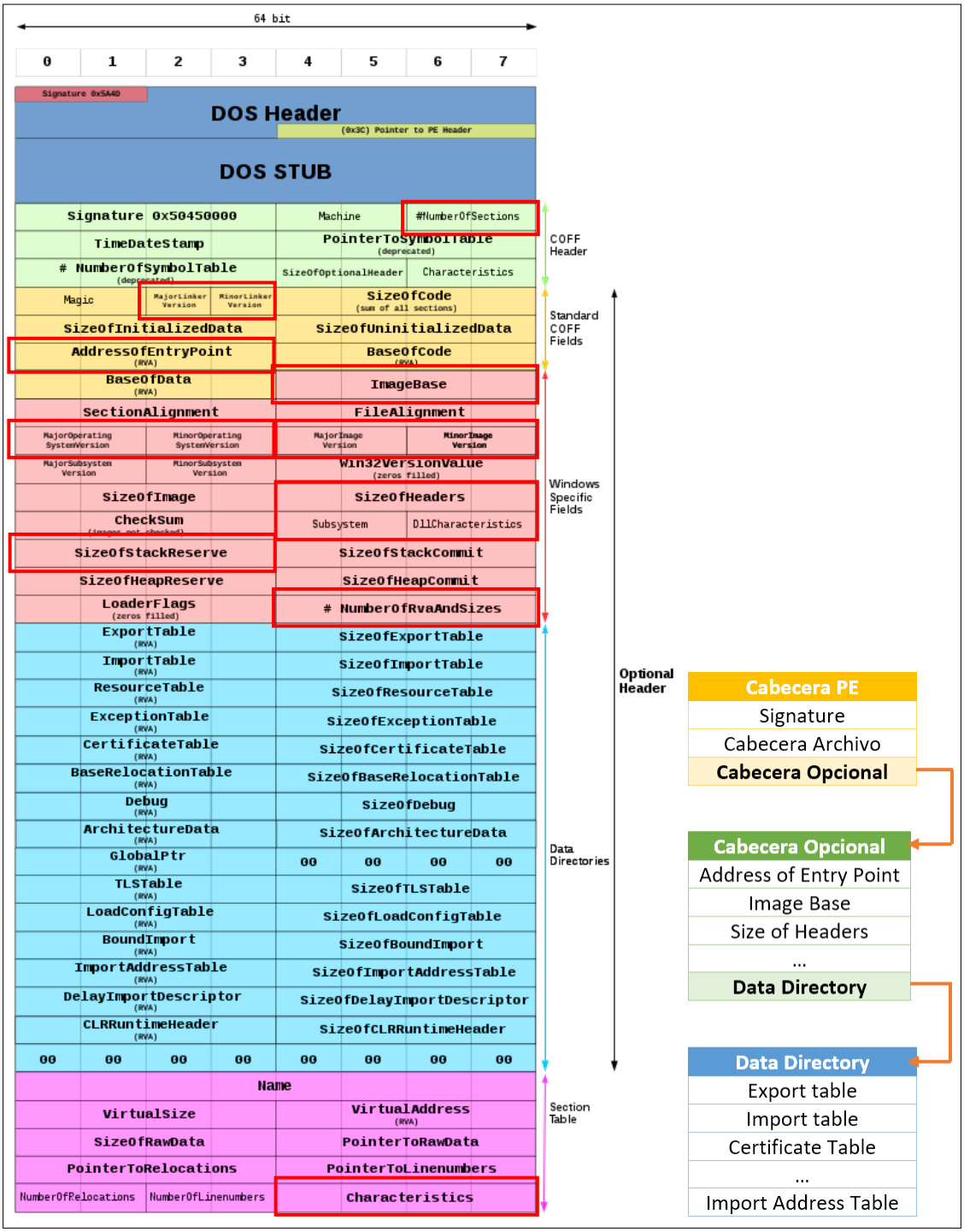


Figura 7.1 A la izquierda, esquema de las secciones de un fichero PE con los campos correspondientes a las características marcados en rojo. Adaptado de Wikipedia (https://ca.wikipedia.org/wiki/Fitxer:Portable_Executable_32_bit_Structure_in_SVG_fixed.svg) CC BY 4.

La figura 7.1 marca cuáles son las características comentadas y su situación en cada una de las secciones de la estructura PE. También se explica esquemáticamente la relación entre la cabecera principal, la cabecera opcional y el contenido del directorio de datos.

7.2- Código e Implementación desarrollado

Los scripts y código desarrollado están escritos en Python 3 y utilizan librerías para el tratamiento de archivos de tipo Comma-Separated Values (CSV) y Scikit-learn en su versión 1.0.1.

SeleccionadorCaracteristicas.py

Lee los archivos CSV con características del subdirectorio *./dataset/* y, en función de la cantidad de ejecutables maliciosos y no maliciosos especificados para ser recuperados, los almacena temporalmente en una lista. Posteriormente, lee las columnas del archivo de texto *características*UtilizadasUtilizadas.txt que coinciden con las características que se van a usar para este trabajo de fin de máster y vuelca solamente esas en el archivo *caracteristicasFinales.csv*

El script está construido para dar flexibilidad a que se pueda modificar el conjunto de características recuperadas del dataset original, simplemente añadiendo o quitando atributos (nombres de columnas) de *características*UtilizadasUtilizadas.txt.

SeleccionadorCaracteristicas.py hace uso de *LineaFeaturizer.py* que crea un objeto con los 90 atributos de cada línea correspondiente a un binario del set de datos para luego devolverlo como un diccionario de datos y que sea así fácilmente manejable por el script *SeleccionadorCaracteristicas.py*

Admite los parámetros:

- `-b <número de binarios benignos>` para la cantidad de binarios no maliciosos que se van a recopilar del dataset.

- `-m <número de binarios maliciosos>` para la cantidad de binarios maliciosos que se van a recopilar del dataset.

CorrelacionesAltas.py

Este script genera una gráfica en la cual se calcula la correlación entre las variables. Aunque es cierto que la mayoría de las características se han escogido siguiendo criterios de ciertas publicaciones científicas, los datos con los que se trabaja pertenecen a binarios no maliciosos ransomware muy variados. Esto podría hacer que hubiera una fuerte correlación entre ellas e indicaría que algunos datos son redundantes y poseen información muy similar.

Un detalle del código que realiza este procedimiento está escrito en la figura 7.2. Primero se calcula la correlación en función de los datos del archivo de resultante en el script anterior *caracteristicasFinales.csv* (aunque se puede usar cualquier otro fichero similar), eliminando la columna *"Malicious"* que simplemente indica si el ejecutable es ransomware o no y, de no eliminarla, alteraría significativamente los resultados puesto que sus valores son "0 "para archivos no maliciosos y "1" para los maliciosos.

Al final se genera la gráfica como un mapa de color. Examinando la gráfica se podrá tantear un "umbral" que servirá para eliminar características que no interesen y así poder reducir la dimensión de los datos en los scripts de entrenamiento para generar el modelo.

Admite el parámetro:

- `-f <fichero con características>` Especifica el fichero con características con el que trabajará

```
#Calculando correlación
 booleanos = [i for i in data.columns if 'bool' in str(data.dtypes[i])]
 enteros = [i for i in data.columns if 'int' in str(data.dtypes[i]) or
'float' in  str(data.dtypes[i])]
 corr = data[enteros + booleanos].corr()

#Eliminando columnas según un umbral definido
 upper = cormat.abs().where(np.triu(np.ones(cormat.shape),
k=1).astype(np.bool_))
```

Figura 7.2 Detalle del código empleado para el cálculo de la correlación en CorrelacionesAltas.py y la eliminación de las columnas altamente correlacionadas en RandomForest_model.py.

RandomForest_model.py

Aquí se implementa todo el proceso de generación del modelo mediante entrenamiento con Random Forest.

En primer lugar, se carga en memoria el archivo con las características y el script elimina las características altamente correlacionadas según un umbral. El nombre del archivo y el umbral se debe especificar en la línea de comandos a la hora de iniciar el script, al igual que todos los parámetros propios de Random Forest que se considere oportuno incluir. De no incluir esto último, se tomarán los valores por defecto indicados en la documentación de Scikit-Learn sobre este algoritmo de clasificación en concreto.

En segundo lugar, el hiperparámetro *class_weight* debe especificarse, ya que no es recomendable dejar un valor por defecto como "balanced" o "balanced_subsample". Por lo tanto, es necesario el uso de la función *compute_class_weight* que, en función del número de elementos de cada clase, calculará las proporciones adecuadas.

En tercer lugar, se realiza un proceso de estandarización con el fin de reescalar los datos.

En cuarto y último lugar, una vez eliminadas las correlaciones altas y reescalado los datos, el código procede a entrenar el modelo, generar la matriz de confusión, una gráfica con la curva ROC y volcarlo en el archivo *RF_Modelo.mdl*

Admite los parámetros:

- `-f <fichero con características>` Especifica el fichero con características con el que trabajará.

- `-co <Umbral de correlación>` Un valor estimado en base al mapa de color creado por *CorrelacionesAltas.py*

- Todos los parámetros de Random Forest que permite la librería.

Es relevante aclarar que, finalmente, se optó por modificar el script *SeleccionadorCaracteristicas.py* para que sustituyera los valores booleanos que indican si un

binario es benigno o ransomware por una “B” y una “M” en función del tipo para evitar confusiones.

Xgboost_model_sklearn.py

Funciona exactamente igual que en *RandomForest_model.py* solo que se implementa el algoritmo XGBoost.

Tanto Random Forest como XGBoost incorporan código comentado con el que se realizó Cross-Validation de forma manual con el propósito de encontrar la mejor configuración de hiperparámetros. Para XGBoost este código es algo desigual ya que incorpora la función “cv” que ya realiza el Cross-Validation para cada iteración de manera optimizada y además evita recurrir a la librería *kfold* de Scikit-Learn.

Comprobador_ransomware.py

Es el script que se utiliza para comprobar el funcionamiento del modelo. Primero recibe la ruta del ejecutable y llama al script *Static_Feature_Parser.py*. Se trata de una adaptación del código del procedimiento de extracción de características estáticas incluido en el repositorio de GitHub donde se almacena el dataset empleado. Hace acopio de la librería *pefile* de Python.

Una vez *Static_Feature_Parser.py* extrae la información del binario, lo devuelve a *Comprobador_ransomware* para convertirlas en un diccionario de datos con el que poder trabajar eliminando las características no utilizadas. Antes de generar la predicción se carga la información de reescalado (datosScaler.joblib) utilizados en la creación de los modelos; de no hacerlo así los valores resultantes sobre la predicción serían inexactos.

En la última sección carga el modelo y predice, con los datos estandarizados, si la muestra es ransomware o no junto con la probabilidad.

Admite los parámetros:

- `-e <fichero con ejecutable>` Especifica el fichero binario con el que se trabajará.

- `-m <modelo de clasificación>` Para seleccionar el modelo generado por el algoritmo de clasificación.

- `-t < "RandomForest", "XGBoost">` Indica el tipo de algoritmo correspondiente al modelo predictivo.

7.3- Configuración del set de datos y características

Si se quiere generar el fichero CSV con las características de los dos tipos de binarios hay que examinar antes de cuántos de estos ejecutables se dispone. El dataset cuenta con 1700 registros ransomware y 1947 registros no maliciosos. Lo ideal es incluir una cantidad equivalente en ambos para evitar sesgos, sobre todo teniendo en cuenta que el número de características es relativamente pequeño. Con el script desarrollado que selecciona y prepara las características (SeleccionadorCaracteristicas.py) se puede crear un archivo rápidamente con el número deseado.

```
python SeleccionadorCaracteristicas.py -b 1700 -m 1700
 Archivo procesado-----> ./dataset/Dataset.csv
 Archivo creado con 3400 registros -----> caracteristicasFinales.csv
```

Tal y como se ha explicado anteriormente, los binarios maliciosos quedan identificados con la letra 'M' en la columna "Malicious" y los que no, con la letra 'B'. Esta columna se omite del análisis de la correlación y el entrenamiento del modelo, pues no aporta valor más allá de servir para clasificar entre benignos o maliciosos el juego de entrenamiento, el de los test y los resultados

Cuando se trabaja con un algoritmo de clasificación resulta imprescindible dividir los datos en dos grupos: set de entrenamiento y set de pruebas. Tanto para Random Forest como para XGBoost la proporción es del 80% (2.720 muestras) y 20% (680 muestras) respectivamente, además de ser aleatoria en cada ejecución del script. La división y aleatoriedad se consigue con llamando al siguiente método:

```
train_test_split(x, y, test_size=0.20, shuffle = True)
```

Donde *test_size* especifica el tamaño del conjunto de pruebas (un 20%) quedando el resto para el entrenamiento y *shuffle* hace que los datos se barajen aleatoriamente antes de dividirse. La decisión viene motivada porque el número de ejecutables dentro del set de datos no es grande y, en tal caso, se hubiese optado por un valor porcentual algo mayor que 20.

7.3.1- Parámetros de evaluación del modelo

Los modelos se evaluarán obteniendo una serie de métricas, que son las siguientes:

- Matriz de confusión: Define cómo el algoritmo a clasificado cada uno de los conjuntos de datos en verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. Proporciona información sobre las expectativas del modelo a la hora de predecir, en este caso, el tipo de binario ('M' o 'B'). En la tabla 7.2 puede consultarse un ejemplo real de lo que representa este tipo de matriz y, además, correspondiente al modelo de árbol de decisión entrenado en este trabajo de fin de máster.

- Puntuación de exactitud: La puntuación de exactitud (accuracy) representa el número de instancias de datos clasificadas correctamente sobre el número total de predicciones. El cálculo se realiza sobre lo obtenido en la matriz de confusión usando la fórmula: `(Verdaderos Positivos + Verdaderos Negativos) / (Verdaderos Positivos + Verdaderos Negativos + Falsos Positivos + Falsos Negativos)`. También la tabla 7.2 posee un ejemplo del cálculo.

- Recall: Indica los resultados correctos (Verdaderos Positivos y Verdaderos Negativos) que proporciona el modelo sobre los datos del test. Si solo fuese capaz de clasificar correctamente la mitad de los binarios el valor sería del 50%.

- F-score: Responde a la siguiente fórmula que calcula la media armónica entre la precisión y el *recall*: `2 x [(Precisión x Recall) / (Precisión + Recall)]`

7.3.2- Validación cruzada (K-Fold)

La validación cruzada permitirá obtener métricas sobre el desempeño del modelo contrastándolo con los datos reales del set de entrenamiento que, como ya se ha explicado, representará 1/5 del total de muestras. En este trabajo se define un valor *K-Folds* = 5, de forma que se harán cinco entrenamientos independientes del modelo para obtener un promedio de

todas las iteraciones; cada entrenamiento utiliza secciones de los dos conjuntos de datos (entrenamiento y pruebas) diferentes y aleatorios.

Concretamente, el procedimiento seguido consiste en estos pasos:

1. Cargar los datos de los archivos PE en memoria.
2. Dividir los datos en 2.720 para el juego de entrenamiento y 680 para el de pruebas.
3. Entrenar el modelo inicial y que se va a guardar, con los algoritmos Random Forest o XGBoost.
4. Ejecutar la validación cruzada y mostrar los datos de *accuracy* de los cinco entrenamientos.
5. Mostrar las puntuaciones del modelo obtenido en el paso 4.
6. Un indicador positivo sería que las métricas de la validación cruzada y las del modelo inicial fuesen similares.

Si una de las iteraciones solo contuviese muestras de binarios benignos o viceversa, podría predecir siempre, que cualquier fichero ejecutable es benigno o malicioso. Por este motivo se optado por la versión de K-Fold: “Stratified K-Fold” (K-Fold estratificado), pues mantiene un equilibrio en el número de clases y resulta útil cuando el tamaño de los datos es pequeño. (AnalyseUp, s.f).

Una técnica relacionada se empleó para realizar una estimación sobre cuáles son los hiperparámetros más adecuados; fue el *Nested Cross-Validation* a través de *GridSearchCv*. GridSearchCV es un método de ajuste de hiperparámetros, incluido en Scikit-Learn, que permite probar varias combinaciones de valores dentro de un rango de parámetros durante una serie de iteraciones. Puesto que hace uso de *StratifiedKFold*, divide el conjunto de datos de la misma forma que ya se ha explicado.

Así se puede mostrar un resultado exponiendo las mejores combinaciones y precisión de modelo. De cualquier modo, si la cantidad de iteraciones y posibles valores de hiperparámetros es alta, la complejidad aumenta exponencialmente y puede resultar imposible o muy costoso de llevar a cabo. El código utilizado para esto se encuentra comentado en el script *RandomForest_model.py* y el script Xgboost_model_sklearn.py

7.3.3- Estandarización y normalización

En las características seleccionadas existe mucha disparidad entre la magnitud de variables como *SizeOfStackReserve* o *AddressOfEntryPoint* y el resto, por lo que se debe aplicar un proceso de estandarización o normalización porque, de no hacerlo, afectaría a todo el proceso de entrenamiento. Algunos ejemplos de las diferencias de magnitud aparecen en la tabla 7.1. En este caso se ha optado por la estandarización mediante la clase `sklearn.preprocessing.StandardScaler`.

Unos valores atípicos pueden afectar a los resultados especialmente si se usan métodos con cálculos sobre la distancia para la clasificación. Por otro lado, cuando se realiza un reescalado, se debe aplicar a todo el conjunto de datos, tanto de entrenamiento como de prueba, así que este proceso se incorpora en el script que se encarga de entrenar el modelo antes de realizar divisiones entre las variables `X_train` y `X_test`. La estandarización mejoró ligeramente el rendimiento de los dos algoritmos.

Los datos de reescalado se deben almacenar para emplearlos también en la información recogida del binario que sobre el que se va a predecir si su comportamiento corresponde con el ransomware o no.

MajorLinkerVersion	NumberofSections	SizeOfStackReserve	Characteristics	AddressOfEntryPoint
7	8	1048576	271	150944
6	5	4194304	258	12538
…	…	…	…	7680
6	3	262144	271	4432
9	6	327680	302	278172
6	3	1048576	271	5824

Tabla 7.1 Ejemplo acortado de las diferencias de magnitud entre algunas variables antes de aplicar StandardScaler de Sklearn.

El siguiente paso es calcular las características altamente correlacionadas. Para ello se lee el archivo CSV resultante del anterior script: “caracteristicasFinales.csv”. La gráfica resultante de la figura 7.3 ayudará a establecer un umbral adecuado que elimina las características altamente correlacionadas.

```
python CorrelacionesAltas.py -f caracteristicasFinales.csv
```

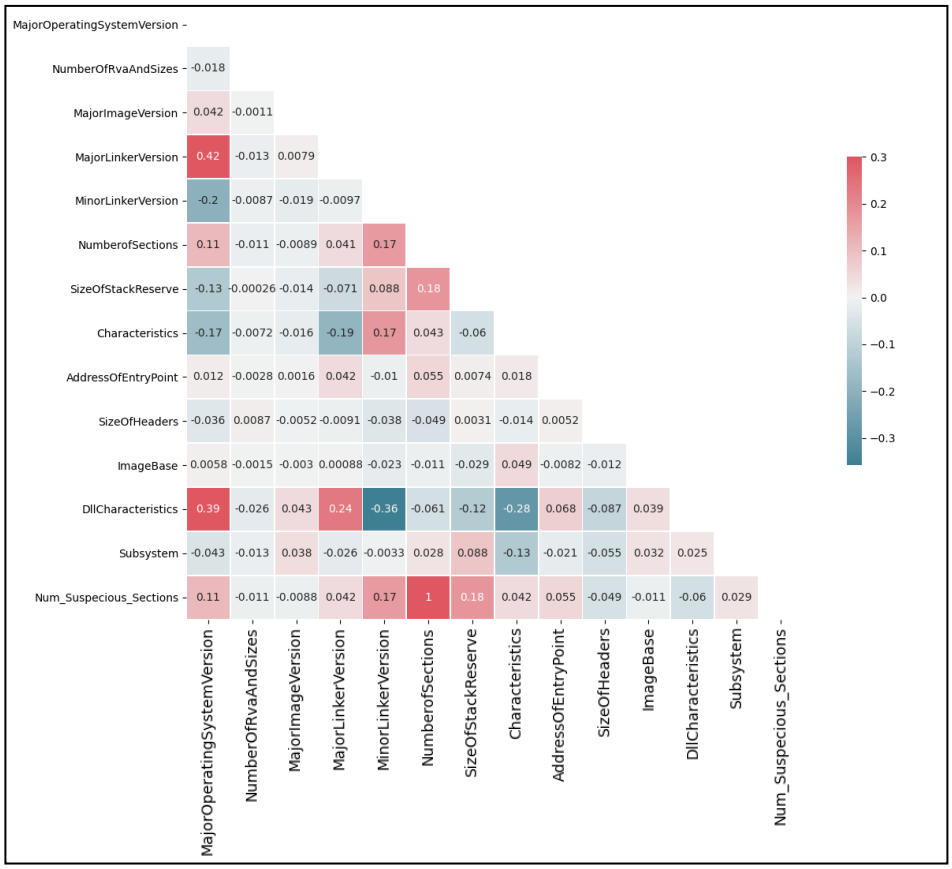


Figura 7.3 Gráfica con características con una fuerte correlación.

Se muestra como existe una correlación fuerte a partir de -0.35 y 0.35. Entre las características más llamativas destacan *MajorLinkerVersion*, *DllCharacteristics* y *Num_Suspecious_Sections*:

- MajorLinkerVersion es bastante similar a *MajorOperatingSystemVersion*, pero el origen de los datos con respecto a los archivos ejecutables no tiene relación. Podría considerarse su eliminación y comprobar si tiene algún efecto sobre el modelo.

- Num Suspecious Sections también tienen una correlación muy alta con *NumberofSections*. Esto tiene sentido al entender que las secciones no sospechosas que abarca el set de datos son solo las principales que debería tener cualquier fichero malicioso por lo que el número total de secciones y el número de secciones sospechosas es muy similar.

- DllCharacteristics cuyo contenido no tiene ninguna relación con *MajorOperatingSystemVersion* y define cómo se inicializarán otras librerías DLL, por lo que podría ser poco recomendable eliminarlo.

Teniendo en cuenta esta información se puede definir un umbral de 0.35 para que se incluya *DllCharacteristics*, *MajorLinkerVersion* y *Num_Suspecious_Sections*.

7.4- Pruebas y resultados obtenidos con RandomForest

Como primer algoritmo seleccionado, se entrena el modelo con RandomForest.

Haciendo balance de las características más y menos importantes, ejecutando una prueba con los hiperparámetros por defecto, se obtiene la gráfica visualizada en la figura 7.4.

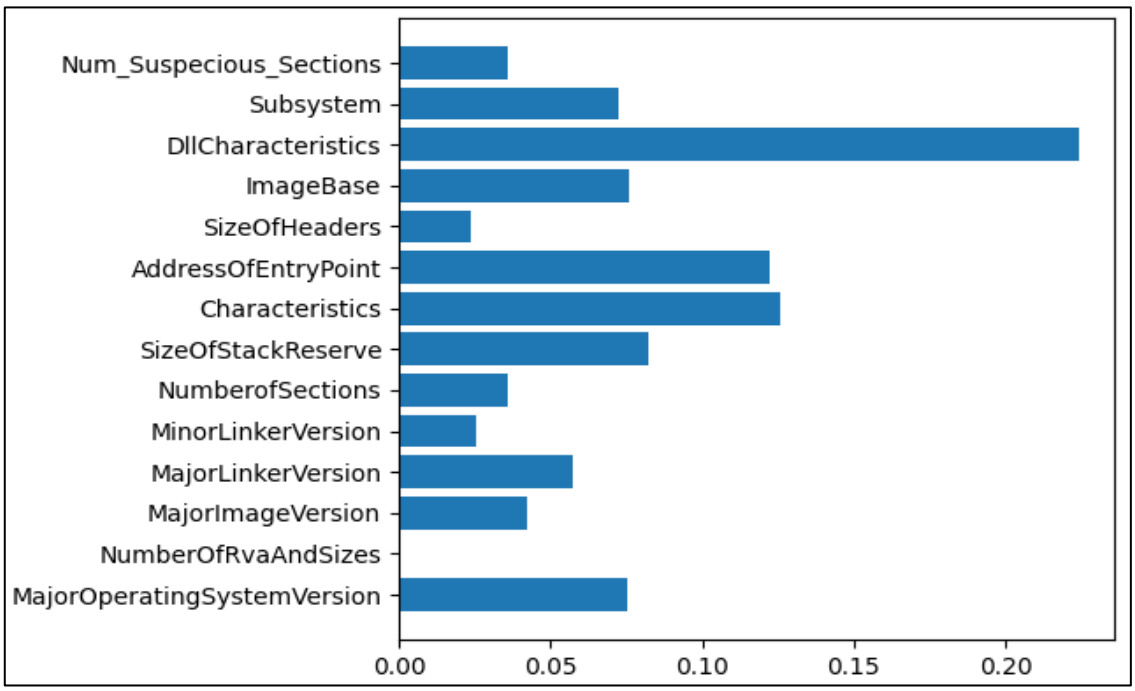


Figura 7.4 Gráfica con la importancia de todas las características del dataset empleado para Random Forest.

La figura 7.4 expone la importancia de cada característica, incluidas las que se han eliminado por la correlación. En pruebas anteriores con peores resultados se habían eliminado algunas bastante relevantes.

DllCharacteristics es la propiedad con más peso para el modelo.

NumberOfRvaAndSizes no tiene apenas relevancia y podría descartarse totalmente del conjunto de datos. La mayoría de los binarios analizados poseen una cantidad similar de elementos en el directorio de datos, así que no constituye diferencias suficientes entre un archivo ejecutable normal y uno ransomware.

También es llamativo que *MinorLinkerVersion* tenga menos relevancia que *MajorLinkerVersión*. Examinando los datos, esto podría deberse a que muchos binarios conserven una versión mínima del enlazador con un valor de 0. Sin embargo, *MajorLinkerVersion* era una de las candidatas a su exclusión en el entrenamiento del modelo por presentar una fuerte correlación, sobre todo con *characteristics*, *Num_Suspecious_Sections y MajorOperatingSystemVersion* que, además, no tienen ninguna relación en el origen de los datos.

Después de varias pruebas y entrenamientos, la mejor configuración de hiperparámetros fue la siguiente:

- Un umbral de 0.40. Haciendo pruebas con umbrales más bajos donde se eliminan más características, la curva ROC y la puntuación de exactitud dejaban que desear.

- n_estimators entre 100 y 200. Este hiperparámetro controla la cantidad de árboles aleatorios que se generarán dentro del clasificador. Aunque el valor ideal es 60, un número cercano a 150 mejoró la eficiencia del modelo. En la figura 7.6 de los resultados de XGBoost se puede observar que con *Nested Cross-Validation* se llegó a la conclusión de que 60 es el valor óptimo y proporciona muy buenos resultados en el algoritmo. No obstante, en lo que respecta a RandomForest, se observó que era más adecuado una cantidad al menos el doble de alta.

- Bootstrap con valor verdadero (True). Esto provocará que el algoritmo use un tamaño de muestra igual al conjunto original.

- class_weight con el resultado del cálculo provisto por la función compute_class_weight. Aunque no se especifica en línea de comandos, si que se asigna internamente en el código del script con el valor *balanced_subsample*

- Max_features controla el número máximo de características que el algoritmo prueba en cada árbol. Se ha observado que con *sqrt* la puntuación de exactitud casi siempre es mayor de 0.955 (95,5%) por lo que se escoge este método para la cantidad máxima.

- Atendiendo a las observaciones previas en la gráfica de la figura7.3 y en la figura 7.4, las características excluidas fueron:
    - *NumberOfRvaAndSizes* por presentar nula relevancia.
    - Num_Suspicious_Sections por presentar una fuerte correlación.

Debido también a la importancia de las características que se muestran en la gráfica de la figura 7.4, los más coherente es no excluir las características *DllCharacteristics* y *MajorLinkerVersión*.

7.4.1- Incluyendo mejor selección de hiperparámetros y características

La ejecución del script para entrenar el modelo que proporcionó mejores resultados y la salida en la consola de comandos, según las observaciones detalladas en el apartado anterior, es:

```
python RandomForest_model.py -f caracteristicasFinales.csv -co 0.45 -
n 150 -b True -m sqrt
```

```
Eliminando columna altamente correlacionada:
Num_Suspecious_Sections...
2720 samples in training set, 2 labels in training set
680 samples in testing set, 2 labels in testing set
_____________________Validación Cruzada_____________________
Puntuaciones de Stratified Cross Validation
[0.96507353 0.97794118 0.96691176 0.95036765 0.96323529]
Puntuación media de Stratified Cross Validation
0.96 de exactitud con una desviación estándar de 0.01
_____________________Puntuaciones del modelo_____________________
Matriz de confusión:
[[400  16]
 [ 16 418]]
Puntuación de exactitud(accuracy) en el test
0.9623529411764706
```

```
Reporte de clasificación
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       416
           1       0.96      0.96      0.96       434

    accuracy                           0.96       850
   macro avg       0.96      0.96      0.96       850
weighted avg       0.96      0.96      0.96       850
```

La mayoría de los intentos de entrenamiento superaron el 95% de exactitud y se situaban entre **≈0.95 y ≈0.962** siendo ese el mejor de los resultados. Las pruebas de validación cruzada se sitúan casi en la totalidad de ejecuciones del script en un promedio **= 0.96**, lo cual coincide con el *accuracy* del modelo generado y es señal de que está funcionando bien. La exactitud se calcula en función de la tasa de verdaderos positivos, falsos positivos, falsos negativos y verdaderos negativos. Esto se puede ver en la matriz de confusión, que representa lo expuesto en la tabla 7.2.

Verdaderos Positivos (TP) Realmente era ransomware y se detectó 411	**Falsos positivos (FP)** Realmente no era ransomware y se detectó como ransomware. 17
Falsos negativos (FN) Realmente era ransomware y no se detectó como ransomware 8	**Verdaderos negativos (TN)** Realmente no era ransomware y no se detectó como ransomware 414
$\text{Exactitud} = \frac{TN + TP}{\text{TN} + \text{TP} + \text{FP} + \text{FN}} = \frac{411 + 408}{411 + 408 + 15 + 16} = 0.963529$	

Tabla 7.2 Explicación de la matriz de confusión y el cálculo de la exactitud en Random Forest.

Estas puntuaciones son interesantes porque el número de falsos negativos es bajo y, de cara a implementar un sistema de detección / prevención de intrusiones para ransomware, siempre será mejor que detecto un fichero ejecutable como falso positivo (cuya tasa es relativamente alta) que como falso negativo y tenga lugar una infección en los sistemas informáticos.

El resto de los valores correspondientes al reporte tienen el siguiente significado:

- Recall: La medida hace balance de cuántos elementos correctamente clasificados de la clase (0 o 1) hay respecto al número total de elementos de esa misma clase. Indica que las dos clases están muy equilibradas y, puesto que los valores son altos, sugiere que los resultados del clasificador son precisos. 0 sería la clase correspondiente a los ficheros ejecutables que no son malware y 1 los que corresponden a ransomware.

- F1-Score: Es un promedio entre la exactitud (accuracy) y la puntuación Recall. Los valores están comprendidos entre 0 y 1, donde 0 es la peor puntuación y 1 la mejor. En este caso, lógicamente la media de las dos puntuaciones es la misma puesto que ambas tienen el mismo valor.
- weighted avg: Muestra un promedio sobre los pesos asignados a las clases del dataset. Este promedio resulta ser tan preciso como los valores de exactitud o Recall e indica que ambas estaban muy equilibradas.

La curva ROC ahora queda plasmada en la figura 7.5:

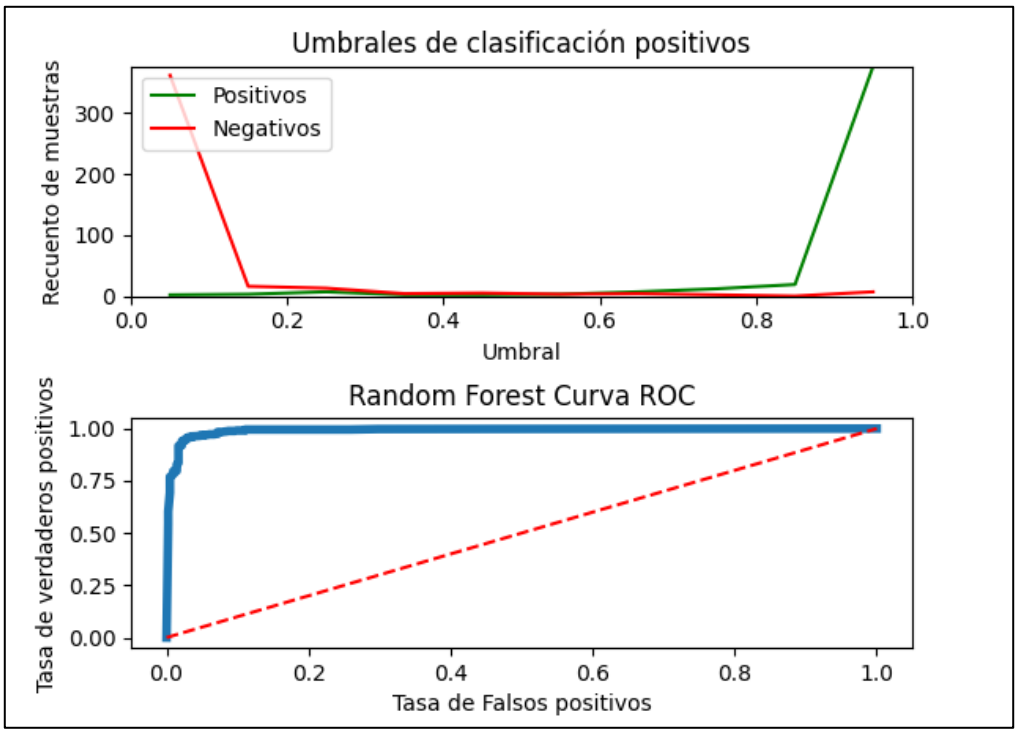


Figura 7.5 Curva ROC con el mejor resultado de entrenamiento de Random Forest.

7.4.2- Descartando mejor selección de hiperparámetros y características

En otro escenario distinto, con un umbral más bajo (entre 0.15 y 0.30) donde se eliminan por su alta correlación las características: *NumSuspeciousSections*, MajorLinkerVersion, DllCharacteristics; el no cálculo del peso de las clases (*compute_class_weight*), un número máximo de características basado en el cálculo del logaritmo: *log2* donde antes era *sqrt* y un valor para *n_estimators* menor (el valor óptimo calculado con *GridSearchCV()*), los valores son mucho peores:

```
python RandomForest_model.py -f caracteristicasFinales.csv -co 0.30
-n 60 -b True -m sqrt
Eliminando columna altamente correlacionada: MajorLinkerVersion,
DllCharacteristics, Num_Suspecious_Sections....
---Creando los sets y entrenando el modelo---
2720 samples in training set, 2 labels in training set
680 samples in testing set, 2 labels in testing set
_________________Validación Cruzada_________________

Puntuaciones de Stratified Cross Validation
[0.94669118 0.95036765 0.95588235 0.96139706 0.95588235]
Puntuación media de Stratified Cross Validation
0.95 de exactitud con una desviación estándar de 0.01
_________________Puntuaciones del modelo_________________
Matriz de confusión:
[[419  26]
 [ 22 383]]
Puntuación de exactitud (accuracy) en el test
0.9435294117647058
Reporte de clasificación
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       445
           1       0.94      0.95      0.94       405
```

```
    accuracy                           0.94       850
   macro avg       0.94      0.94      0.94       850
weighted avg       0.94      0.94      0.94       850
```

La información extraída de la salida por pantalla denota una exactitud más baja junto con el resto de los valores descritos anteriormente y un número de falsos negativos y falsos positivos poco aceptable. Tiene sentido, pues el peso de las características ya visto al principio de este apartado, corresponde con algunas muy importantes que sí se han eliminado.

Del mismo modo, también decae la media de *accuracy* en la validación cruzada un punto respecto al apartado de mejor selección de hiperparámetros y características. Para terminar, el mejor modelo es guardado con el nombre *RF_Modelo.mdl* para su posterior utilización.

7.5- Pruebas y resultados obtenidos con XGBoost

El segundo algoritmo que se va a utilizar es XGBoost en su implementación con la API de Scikit-Learn: "XGBClassifier".

Como en cualquier otro algoritmo de clasificación, pero para este algoritmo de forma destacada, hay un número significativo de hiperparámetros que se pueden configurar, sin embargo, los que más relevancia presentaron a la hora de afinar los resultados fueron los que se describen en este apartado. Además, se obtuvieron mediante las técnicas de *Nested Cross-Validation* mencionados en el algoritmo Random Forest:

- n_estimators define el número de rondas máximas en las que se van consiguiendo mejoras en la precisión del algoritmo progresivamente. Se alcanza por lo tanto un punto óptimo en el que, a partir de ahí, como cada nueva ronda intenta adaptar y corregir los errores cometidos en las anteriores iteraciones, continuar con nuevas rondas resulta contraproducente y empeora los resultados. En el código del script se incorpora una sección para calcular el valor adecuado para este dataset y el mejor es 60. Puede comprobarse en la figura 7.6.

```
#Calculando el mejor n_estimators----------------
#Se van a evaluar una serie de valores de 10 a 200 de 10 en 10
n_estimators = range(10, 200, 10)
param_grid = dict(n_estimators=n_estimators)
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=7)

buscador_estimators = GridSearchCV(XGBClassifier(use_label_encoder=False),
          param_grid,scoring="neg_log_loss", n_jobs=-1, cv=kfold)
grid_result = buscador_estimators.fit(X_train, y_train,eval_metric='rmse')

print("Mejor valor: %f usando un número %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) con: %r" % (mean, stdev, param))
```

```
Mejor valor: -0.118999 usando un número {'n_estimators': 60}
-0.119783 (0.024156) con: {'n_estimators': 50}
-0.118999 (0.023635) con: {'n_estimators': 60}
-0.119014 (0.023820) con: {'n_estimators': 70}
……
-0.131360 (0.029412) con: {'n_estimators': 180}
-0.132175 (0.029572) con: {'n_estimators': 190}
```

Figura 7.6 Fragmento de código de la sección que calcula el número óptimo de rondas (n_estimators) y la salida por pantalla.

- max_depth. Puesto que XGBoost realmente trabaja con árboles de decisión aquí se especifica la profundidad de cada árbol. El mejor valor testeado está entre 25 y 30.

- min_child_weight define la suma mínima de los pesos que tendrán que tener las características en un nodo hijo de un árbol. Entre los distintos rangos probados el mejor valor es 1.

- Gamma condiciona cuándo se realizará una división en el nodo de un árbol definiendo la reducción de pérdida más baja a partir de la cual esta división sucede. Finalmente, el mejor valor es 0.2.

- Subsample indica la cantidad de datos que serán muestreados aleatoriamente para usar en cada árbol. Si el valor fuera de 0.5 significaría que solo usaría la mitad. Los valores típicos se mueven entre 0.5 y 1. Para este set de datos y características escogidas el mejor valor es 0.8.

- colsample_bytree y colsample_bylevel definen la proporción de submuestras de columnas al crear cada árbol y cada nivel de un árbol respectivamente. Al igual que en el hiperparámetro subsample los valores típicos están entre 0.5 y 1 y también el mejor valor para los dos es 0.7 o 0,8

7.5.1- Incluyendo mejor selección de hiperparámetros y características

La ejecución del script para entrenar el modelo que proporcionó mejores resultados, junto con su salida en la consola de comandos, fue:

```
python Xgboost_model_sklearn.py -f caracteristicasFinales.csv -co
100 -n 60 -l 0.2 -d 28 -mc 0 -ct 0.8 -cl 0.8 -g 0.1 -su 0.8
---Creando los sets y entrenando el modelo---
2720 muestras en set de entrenamiento,  2 etiquetas en set de
entrenamiento
680 muestras en set de pruebas,  2 etiquetas en set de pruebas
```

```
____________________Validación Cruzada____________________
Puntuaciones de Stratified Cross Validation
[0.96507353 0.95772059 0.96691176 0.97058824 0.96691176]
Puntuación media de Stratified Cross Validation
0.97 de exactitud con una desviación estándar de 0.00
____________________Puntuaciones del modelo____________________
Matriz de confusión:
[[401  18]
 [  4 427]]
Puntuación de exactitud (accuracy) en el test
0.9741176470588235
Reporte de clasificación
              precision    recall  f1-score   support

           0       0.99      0.96      0.97       419
           1       0.96      0.99      0.97       431

    accuracy                           0.97       850
   macro avg       0.97      0.97      0.97       850
weighted avg       0.97      0.97      0.97       850
```

La mayoría de los intentos de entrenamiento superaron el 96,6% y se situaban entre **≈0.965 y ≈0.975** siendo ese el mejor de los resultados. La media en la validación cruzada puede llegar a ser **= 0.97**, indicador de que el modelo está funcionando bien. Por otro lado, la puntuación de exactitud (accuracy) mejoraba si no se eliminaban las características mostradas anteriormente como altamente correlacionadas.

La puntuación es interesante, pues el número de falsos positivos es más alto que en el modelo de Random Forest. Alcanza una exactitud para los binarios ransomware de casi 100%, siendo la tasa de falsos negativos, es decir, ransomware que no se detectó como ransomware, solamente de 4 (en la matriz de confusión) y teniendo un *accuracy* para la clase con binarios no maliciosos del 96%.

La curva ROC de la figura 7.7 incorpora la matriz de confusión y los datos del entrenamiento para XGBoost de manera gráfica:

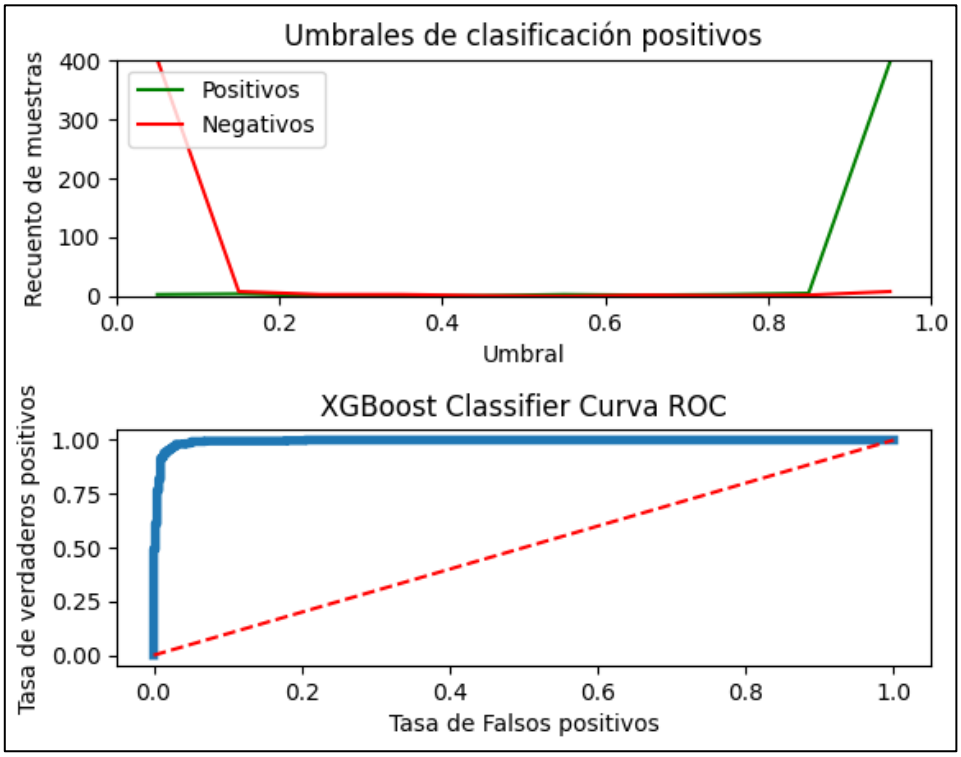


Figura 7.7 Curva ROC con el mejor resultado de entrenamiento de XGBoost.

7.5.2- Descartando mejor selección de hiperparámetros y características

Es destacable la importancia de determinar correctamente el hiperparámetro *n_estimators* al menos en XGBoost. En ciertos artículos de revistas online sobre Machine Learning se establece un valor promedio para XGBoost de entre 100 y 200, aunque esto no deja de ser una decisión subjetiva al tipo y tamaño de datos con el que se está tratando, no se debe tomar la decisión en base a criterios de otros autores.

Ejecutando el algoritmo con *n_estimators* no optimizado, por ejemplo 140 (que es casi el valor utilizado en RandomForest) y descartando, solamente por la alta correlación que presentan, las características *NumberofSections*, *MajorLinkerVersion*, *DllCharacteristics*; Las predicciones del modelo no tienden a ser tan precisas, como queda patente en la matriz de confusión, puntuación de exactitud y, gráficamente, en la figura 7.8.

```
python Xgboost_model_sklearn.py -f caracteristicasFinales.csv -co
0.30 -n 140 -l 0.2 -d 28 -mc 0 -ct 0.8 -cl 0.8 -g 0.1 -su 0.8
Eliminando columna altamente correlacionada: MajorLinkerVersion,
DllCharacteristics, Num_Suspecious_Sections
---Creando los sets y entrenando el modelo---
2720 muestras en set de entrenamiento,  2 etiquetas en set de
entrenamiento
680 muestras en set de pruebas,  2 etiquetas en set de pruebas
_________________Validación Cruzada_________________
Puntuaciones de Stratified Cross Validation
[0.95588235 0.95772059 0.94852941 0.95036765 0.97610294]
Puntuación media de Stratified Cross Validation
0.96 de exactitud con una desviación estándar de 0.01
_________________Puntuaciones del modelo_________________
Matriz de confusión:
[[400  12]
 [ 24 414]]
Puntuación de exactitud (accuracy) en el test
0.9576470588235294
Reporte de clasificación
              precision    recall  f1-score   support

           0       0.94      0.97      0.96       412
           1       0.97      0.95      0.96       438

    accuracy                           0.96       850
   macro avg       0.96      0.96      0.96       850
weighted avg       0.96      0.96      0.96       850
```

Al igual que ocurrió con las pruebas en Random Forest, las métricas en la validación cruzada bajan un punto respecto a la mejor implementación del algoritmo y, tal y como era de esperar, la exactitud final del modelo es más baja, aproximándose al entrenamiento con Random Forest.

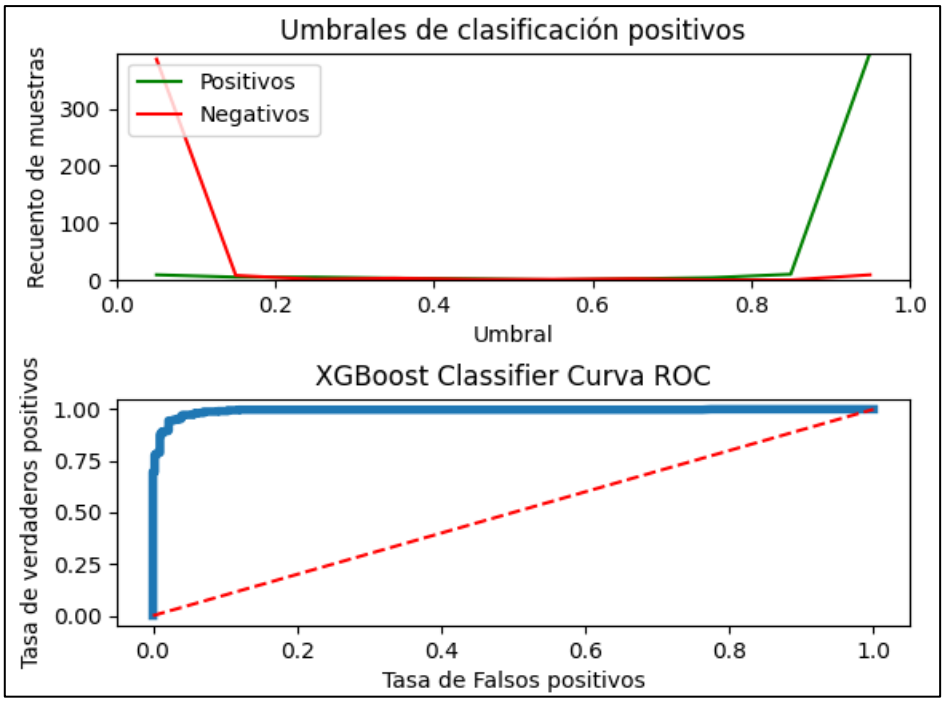


Figura 7.8 Curva ROC descartando mejor selección de hiperparámetros y características con XGBoost.

7.5.3- Comparación respecto a RandomForest

Existe una variación en cuanto a las características que XGBoost concede más o menos importancia con respecto a RandomForest. La figura 7.9 es una muestra de ello.

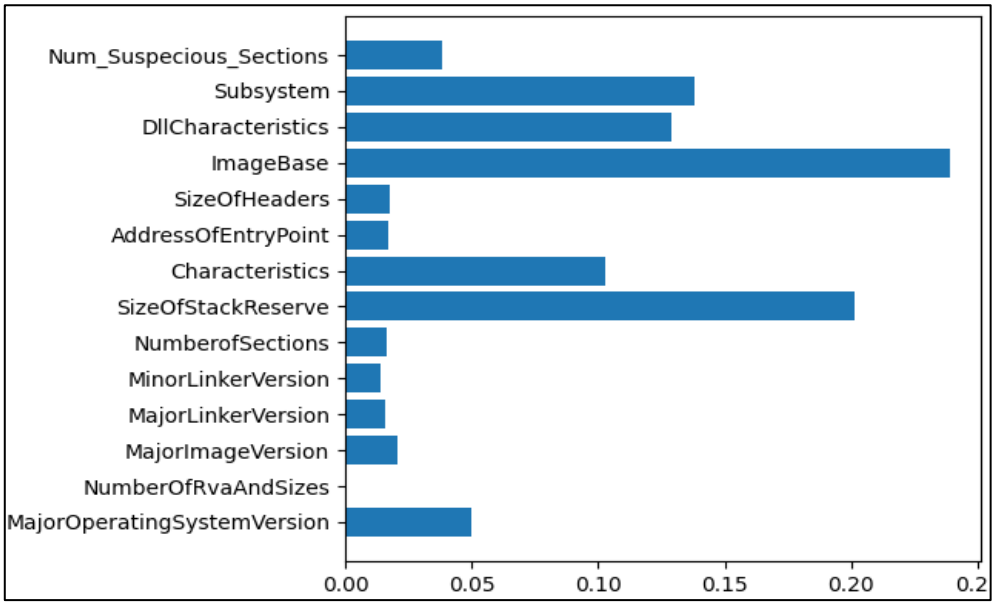


Figura 7.9 Gráfica con la importancia de las características del dataset empleado en Random Forest.

El algoritmo otorga poca relevancia a las características de los campos estándar del encabezado opcional para COFF, que son los relacionados con las versiones del enlazador *MajorLinkerVersion*, *MinorLinkerVersion*, el punto de entrada *AddressOfEntryPoint* y *SizeOfHeaders*. Este último campo también se calcula según el tamaño de, precisamente, este encabezado opcional: Common Object File Format.

Las características que XGBoost sí consideran relevantes son en su mayoría las correspondientes a la cabecera de Windows-Specific como *ImageBase* (cuya notoriedad es destacable), el subsistema imprescindible para la ejecución del binario *Subsystem* y el tamaño de la pila que se reservará *SizeOfStackReserve*. En comparación con RandomForest hay unas características, las específicas de Windows, que quedan muy por encima de las otras más relacionadas con el encabezado COFF, cuya reducida importancia tiene un valor similar.

XGBoost es más sensible a la hora de discernir y establecer un peso o importancia al número de secciones sospechosas que contemplan los creadores del dataset (*Num_Suspecious_Sections*) y el número de secciones (*NumberofSections*) que, como se había explicado antes, tenían unos valores con bastante similitud.

En general ha sido ligeramente más eficaz a la hora generar modelos con mayores tasas de acierto que RandomForest (observable en las métricas de *Stratified K-Fold* y puntuaciones del modelo), aunque la diferencia entre ambos es escasa. Si se han evaluar las mejores configuraciones de hiperparámetros para cada uno de los dos algoritmos, llama la atención la discrepancia de Random Forest sobre el valor óptimo estimado de *n_estimators*, ofreciendo resultados con una cantidad mayor, cercana al doble.

Terminando las pruebas con XGBoost, el mejor modelo resultante de los entrenamientos llevados a cabo es guardado con el nombre *XGBoost_Modelo.xgb* para su posterior utilización.

7.6- Pruebas con archivos binarios en los modelos predictivos

Con el fin de comprobar, aunque de forma somera, la fiabilidad de los modelos predictivos, se diseña una batería de pruebas de dos tipos: binarios benignos y binarios con malware ransomware. *Comprobador_ransomware.py* imprimirá por pantalla su predicción y las probabilidades asociadas, con dos formatos de comandos para cada modelo.

```
python Comprobador_ransomware.py -e "./Batería de pruebas/Benigno/<<NombreEjecutable.exe>>" -m "./capturas/xgboost/modeloFinal.xgb" -t XGBoost
```

```
python Comprobador_ransomware.py -e "./Batería de pruebas/Benigno/<<NombreEjecutable.exe>>" -m "./capturas/RandomForest/modeloFinal.rf" -t RandomForest
```

7.6.1- Pruebas con binarios benignos

Se escogieron archivos PE que muchos usuarios poseen de forma habitual. Tanto paquetes de instalación como el propio *.exe* del software ya instalado. La tabla 7.3 describe cuáles son los ejecutables de las pruebas y el programa o aplicación a la que pertenecen, junto con el hash SHA-256.

Nombre y versión	Descripción	SHA-256
Archivos ejecutables		
WinRAR.exe (v. 5.40.0)	Compresor de archivos, WinRar	a1290e4f7bedc8c7c0c2519bd005caf7fe0210dd374490016e51ed2381731af1
Spotify.exe (v. 1.1.91.824)	Cliente de música en streaming, Spotify	fb35e7250f51254016d53fb1a39bb247e2206812b9a05a537e1d94109ce81f8e
mpc-be64.exe (v. 1.6.3)	Reproductor de vídeo, Media Player Classic - BE	56f3c9e6cf96ab893686a7602b924dfb55705d5652e0d04be78fc3362df8baac
AutoFirma.exe (v. 1.7.2)	Programa de firma electrónica del ministerio de hacienda	7e6803f85d36a147f6019b603abf842481c77e4696a73f0f79b2ee425f3ce6c5
chrome.exe (v. 114.0.5)	Navegador Google Chrome	91e39a6aff4f259121d6bab8076750616d2f9e8d4b92f755bbdd46bf2f00c441

Winword.exe (v. 16.0.10228)	Programa ejecutable Word (Office 2016)	e1c8c56337088db000f78be73382a1c65c8b99c76c2487827ff6579f393ab7de
Skype.exe (v. 8.58.0)	Software para realizar llamadas y vidiollamadas	da66933270eceba4d2ff6a2edfbd57d1eda8a4b2a9cf596a80478edfdf6e1efb
Photoshop.exe (CC 2019 v. 20.0.5)	Programa de edición de imágenes	9df8e3fc14a551baa898d2958b5498309a537b7d63c5509e4655af6baceefe08
Archivos ejecutables como paquetes de instalación		
SteamSetup.exe (v. 2.10.91.91)	Instalador cliente videojuegos en la nube, Steam	3b616cb0beaacffb53884b5ba0453312d2577db598d2a877a3b251125fb281a1
readerdc64_es_xa_install.exe (v. 2.0.0.625s)	Instalador visor de PDF, AdobeReader	f492b470a1b60a5075cd4ebd5b52fa1274f4292a2c8dbd571af208c5b8690b7c

Tabla 7.3 Descripción y hash de los binarios no maliciosos utilizados para el test

Las predicciones del modelo RandomForest y XGBoost para binarios benignos se detallan en las tablas 7.4 y 7.3 respectivamente.

Random Forest		
WinRAR.exe	[0.9933333 – 0.0066666]	No es ransomware
Spotify.exe	[0.7666666 – 0.2333333]	No es ransomware
mpc-be64.exe	[0.9533333 – 0.0466666]	No es ransomware
AutoFirma.exe	[0.2867981 – 0.7132018]	Es ransomware
chrome.exe	[1. – 0.]	No es ransomware
Winword.exe	[0.9866666 – 0.0133333]	No es ransomware
Skype.exe	[0.8 – 0.2]	No es ransomware
SteamSetup.exe	[0.9733333 – 0.0266666]	No es ransomware
readerdc64_es_xa_install.exe	[0.24 – 0.76]	Es Ransonware
Photoshop.exe	[1. – 0.]	No es Ransomware

Tabla 7.4 Predicciones RandomForest en el test no malicioso.

Deliberadamente se ha escogido un programa ejecutable: *AutoFirma.exe*, que realiza operaciones de firma electrónica en documentos, por lo que emplea métodos criptográficos al igual que el ransomware. *AutoFirma.exe* y el programa de instalación *readerdc64_esxa_install.exe* son detectados como ransomware por Random Forest con puntuaciones no muy distantes. Respecto a este último, existe un análisis en un SandBox online del mismo fichero que revela algunas funciones señaladas como "potencialmente criptográficas" [3].

XGBoost		
WinRAR.exe	[0.9997798 – 0.0002200]	No es ransomware
Spotify.exe	[0.9354889 – 0.0645111]	No es ransomware
mpc-be64.exe	[0.9997423 – 0.0002576]	No es ransomware
AutoFirma.exe	[0.0359655 – 0.9640345]	Es ransomware
chrome.exe	[0.9998657 – 0.0001342]	No es ransomware
Winword.exe	[0.9996881 – 0.0003118]	No es ransomware
Skype.exe	[0.9760927 – 0.0239072]	No es ransomware
SteamSetup.exe	[0.9944826 – 0.0055174]	No es ransomware
readerdc64_es_xa_install.exe	[0.2233212 – 0.7766788]	Es ransomware
Photoshop.exe	[0.9998266 – 0.0001733]	No es ransomware

Tabla 7.5 Predicciones XGBoost en el test no malicioso.

XGBoost genera puntuaciones prácticamente en un extremo u otro (0 o 1), con mayor grado de certeza. El algoritmo muestra, al igual que RandomForest, al software desarrollado por el gobierno de España como potencial ransomware y al instalador de Adobe también como falso positivo, pero con un porcentaje menor de probabilidad.

7.6.2- Pruebas con binarios ransomware

Algunos de los ransomware empleados para las pruebas han adquirido gran popularidad o son de reciente descubrimiento, como Lilith. Se indican en la tabla 7.6 los archivos PE ransomware utilizados, junto con el hash identificatorio de la función resumen SHA-256

Nombre	SHA-256
PE: *.exe* (executable)	
lilith	f3caa040efb298878b99f883a898f76d92554e07a8958e90ff70e7ff3cfabdf5
Lockbit	07a3dcb8d9b062fb480692fa33d12da05c21f544492cbaf9207956ac647ba9ae
pandora	1f172321dfc7445019313cbed4d5f3718a6c0638f2f310918665754a9e117733
Darky Lock	fc28d2eaee1fd3416fe3e0cd4669df3ac178c577e3a8c386b1c34c3146afb8d6
ryuk	40b865d1c3ab1b8544bcf57c88edd30679870d40b27d62feb237a19f0c5f9cd1

sodinokibi	861e2544ddb9739d79b265aab1e327d11617bc9d9c94bc5b35282c33fcb419bc
Screen Lock (WinLock)	b25818cfa65b13e2be6358f5c28dfae35578d72fea8d0120486d8ec6629a1bf4
PXJ	9a4e4211f7e690ee4a520c491ef7766dcf1cc9859afa991e15538e92b435f3a1
stop	1bd1ce92414630da3bbb7a0f5f12d1a28be0cd370466db5fa03934587854fe53
PE: *.DLL* (Dynamic-link library)	
Conti	3db6e8df73f12b6a9fa9adb6ad87b017d530a9d736909338042735ed00a9463b
WannaCry	8f86b42241e81b7461ba45fde82505332e4578663fdd20c2723a99863c34ad67

Tabla 7.6 Nombre y hash de los binarios maliciosos utilizados para la prueba.

Las predicciones del modelo RandomForest y XGBoost para binarios ransomware se detallan en las tablas 7.7 y 7.8 respectivamente.

Random Forest		
lilith	[1. – 0.]	No es ransomware
Lockbit	[0.56 – 0.44]	No es ransomware
pandora	[0.9533333 – 0.0466666]	No es ransomware
Darky Lock	[0.2 – 0.8]	Es ransomware
ryuk	[0.4933333 – 0.5066666]	Es ransomware
sodinokibi	[0.34 – 0.66]	Es ransomware
Screen Lock (WinLock)	[0.56 – 0.44]	No es ransomware
PXJ	[0.4933333 – 0.5066666]	Es ransomware
stop	[0.12 – 0.88]	Es ransomware
Conti	[0.28 – 0.72]	Es ransomware
WannaCry	[0.0133333 – 0.9866666]	Es ransomware

Tabla 7.7 Predicciones RandomForest en el test ransomware.

De los once binarios ransomware analizados por Random Forest tres de ellos son falsos negativos, excepto ScreenLock, considerado un tipo de ransomware denominado Locker o bloqueador. Esta modalidad no encripta los archivos de la computadora, sino que bloquea el acceso a la interfaz de Windows mostrando un mensaje de requerida activación falso. El ransomware más reciente tiende a presentar valores entre las dos clases (Benigno – Ransomware) más próximos o provocar un falso negativo. Por ejemplo: Lilith, Lockbit,

Sodinokibi, Darky Lock y PXJ aparecieron entre el año 2020 y 2022, mientras que otros como Stop, Conti y WannaCry son anteriores a esas fechas.

XGBoost		
lilith	[0.9999313 – 0.0000683]	No es ransomware
Lockbit	[0.1244937 – 0.8755063]	Es ransomware
pandora	[0.9998771 – 0.0001228]	No es ransomware
Darky Lock	[0.2139131 – 0.7860868]	Es ransomware
ryuk	[0.0798674 – 0.9201325]	Es ransomware
sodinokibi	[0.0306209 – 0.9693790]	Es ransomware
Screen Lock (WinLock)	[0.7216233 – 0.2783766]	No es ransomware
PXJ	[0.2233953 – 0.7766046]	Es ransomware
stop	[0.0172515 – 0.9827484]	Es ransomware
Conti	[0.0031034 – 0.9968965]	Es ransomware
WannaCry	[0.0001645 – 0.9998354]	Es ransomware

Tabla 7.8 Predicciones XGBoost en el test ransomware.

El algoritmo XGBoost predice más eficientemente las muestras de binarios produciendo dos falsos negativos en Lilith y Pandora, los mismos que con Random Forest, pero acertando en su pronóstico con Lockbit. La distancia entre la probabilidad de las dos clases es más acusada; también para Winlock, donde tampoco erró en este tipo de malware.

8. Conclusiones

8.1- Conclusiones finales

En este trabajo de fin de máster se han alcanzado los objetivos generales y específicos de cada uno de los temas que se proponían. El estudio que se ha hecho del ransomware, junto con las características y otros aspectos que envuelven a este tipo de amenaza, ha resultado ser sumamente didáctico. Al profundizar en cada uno de los objetivos y apartados propuestos se ha podido abarcar un área muy amplia que va desde la distribución de la amenaza y cómo afecta a los sistemas informáticos de forma individual o dentro de una red, hasta cómo pueden mitigarse sus efectos e, incluso, prevenirlos.

Dentro de los objetivos relacionados con la ocultación y el envío masivo de ransomware, se deduce:

- Empleando la herramienta IExpress, junto con un simple script PowerShell, es posible empaquetar en su interior ransomware con un fichero PDF real, de modo que de forma transparente al usuario puede iniciarse el proceso malicioso, mientras que a este se le muestra un documento que simula ser, en este caso, una factura de una compañía eléctrica. Posteriormente, el paquete de instalación generado puede editarse modificando la sección *resources* o introduciendo caracteres Unicode, tomando la apariencia de un archivo legítimo.

- Pese a qué era identificado por algunos programas antivirus como amenaza, se pudo constatar que utilizando el crypter Hyperion, dichos antivirus eran incapaces de detectar el ransomware Ryuk empaquetado en el binario resultante.

- Los experimentos llevados a cabo con Gophish probaron que, creando un servidor SMTP local, el cliente de correo Outlook podía recibir los emails enviados de forma masiva junto con el archivo ejecutable empaquetado con IExpress, indicando, además, un remitente falso, fruto de la alteración de las cabeceras del mensaje (técnica conocida como email spoofing). Sin embargo, si se empleaba un servidor SMTP de Google u Outlook, Gophish

podía completar con éxito el ataque spear-phishing, pero no era posible la modificación de la dirección del remitente ni el envío del programa malicioso.

Respecto al análisis estático, dinámico y de código del ransomware Ryuk, han podido constatarse estas averiguaciones sobre sus capacidades y comportamiento:

- El análisis estático reveló características con las que se podía constatar con gran certeza que el fichero analizado se trataba de ransomware o podía cifrar archivos. Esto resultó claramente observable en las dependencias y llamadas a librerías concretas, así como en la clave pública RSA embebida dentro del código.

- Las cadenas de texto extraídas de Ryuk proporcionan información fundamental sobre sus funcionalidades: persistencia a través de claves en el registro de Windows, movimiento lateral al incluir rangos de direcciones IP locales o interacción con el sistema operativo detectando textos con diversos comandos de Windows.

- Creo importante destacar que la información obtenida del análisis estático no constituye una evidencia de su funcionamiento real, pues la implementación que realizaron los autores, en algunos casos, o bien no fue del todo correcta o bien no era efectiva. Así mismo, observando algunos de esos errores y los textos del código con referencias a otras familias de ransomware, puede afirmarse que los desarrolladores reutilizaron código fuente de creaciones anteriores, prestando poca atención a su utilidad y buen funcionamiento si no representaba una función importante. Además, el análisis en el depurador revela que existen secciones de código sin utilizar y comprobaciones sin ninguna utilidad posterior.

- En el análisis dinámico quedaron acreditadas algunas de las deducciones en cuanto al comportamiento de Ryuk según los datos del análisis estático: borrado de copias de seguridad, movimiento lateral, lectura/escritura de archivos, cifrado, persistencia y privotar entre otras computadoras de la red.

- Asimismo, se comprobó que el sistema de creación de hilos para moverse entre directorios y encriptar los archivos, es muy eficaz y sorprende por su rapidez. El tipo de cifrado y la

cantidad de hilos generados, supone una gran carga computacional para el sistema que el usuario detectará al verse este ralentizado. También se ha podido averiguar qué tipos y tamaños de archivos cifra el ransomware estudiado, pudiendo evitarlo simplemente modificando la extensión del fichero.

- Por otro lado, el interés que han mostrado los desarrolladores de cara a ocultar el ejecutable malicioso y asegurar la persistencia si el sistema operativo fuese reiniciado, es algo pobre. Solo es necesario eliminar cualquiera de los dos archivos ejecutables creados en la misma carpeta o que por algún tipo de error, quizá restricción, del sistema operativo no pueda agregarse la única entrada al registro de Windows que el ransomware trata de insertar, para impedir la ejecución en el siguiente inicio de sesión. Es más destacable el empeño por impedir la recuperación automática del sistema en el reinicio restándole opciones de recuperación al usuario.

- Como consecuencia de los experimentos realizados, se puede afirmar que las posibilidades de Ryuk para propagarse entre otros equipos de la red son bastante limitadas debido a las propias restricciones de las últimas versiones de Windows. Si el foco de la infección no se encuentra en un domino, es poco probable que pueda haber movimiento lateral con una configuración del sistema normal.

En el estudio sobre los métodos criptográficos de Ryuk y el proceso de cifrado de la información, las conclusiones son las siguientes:

- El desempeño criptográfico de Ryuk es impecable. Éste u otros ransomware que operen de forma similar, imposibilitarán el descifrado de los datos a no ser que se obtenga la clave RSA privada.

- Se ha demostrado como las dos APIs que proporcionan los sistemas operativos de Microsoft para generar claves criptográficas: *CryptGenKey* y *CryptGenRandom*, pueden ser interceptadas mediante técnicas de API Hooking con una aplicación de escritorio para Windows. Pese a ello, de las implementaciones con *CryptGenRandom* no podrán

extraerse las claves generadas al estar dentro del espacio del proveedor de servicios Proveedor de Servicios Criptográficos (CSP).

- En este sentido, el autor considera que, de la misma forma que la herramienta desarrollada puede detener el proceso que llame a la *CryptGenKey*, también podría detener cualquier otro proceso. Sin embargo, es posible que los usuarios de una computadora deban poder realizar operaciones de cifrado de documentos, por lo que en caso de un ciberataque de este tipo, todas las contraseñas generadas podrían almacenarse de forma segura y temporal.

Sobre los resultados en la implementación de algoritmos de clasificación empleando unas determinadas características estáticas, se concluye que:

- No es estrictamente necesario utilizar la totalidad de las características estáticas que pueden extraerse de un archivo PE. Es cierto que la postura del autor con respecto a la cantidad de las características escogida resulta ser algo estricta por su limitado número, pero, aun así, ha podido comprobarse que es viable generar unos modelos predictivos con una tasa de aciertos aceptable.

- Aunque la diferencia entre los algoritmos de clasificación RandomForest y XGBoost es escasa, XGBoost consigue una tasa de verdaderos positivos y verdaderos negativos, ligeramente más alta.

- En última instancia, el alumno considera que el dataset empleado para la investigación contiene un número de análisis sobre archivos ejecutables ransomware y benignos insuficiente. Si el dataset incluyera más datos, los modelos predictivos estarían mejor preparados para predecir con más acierto un mayor rango de familias de ransomware.

Se espera que este trabajo aporte tanto conocimiento como detalles útiles a un analista de malware sobre el ransomware y pueda responder a algunas preguntas sobre cómo podría propagarse, qué se puede esperar de su comportamiento, cómo cifra los archivos y qué se puede hacer para mitigar sus efectos o detectarlo a tiempo.

8.2- Trabajo futuro

En este trabajo de fin de máster pueden plantearse múltiples posibilidades de mejora en las investigaciones, metodología y herramientas desarrolladas, lo que quizá derive en nuevos proyectos y líneas de trabajo. Se expondrán aquí esos posibles trabajos futuros según el orden de los temas tratados.

El crypter Hyperion es eficaz para evitar que el código embebido pueda ser detectado o analizado con facilidad, pero, tal y como se ha expuesto anteriormente, muchos sistemas de detección de amenazas lo catalogan como malicioso. Sería de interés indagar sobre otros sistemas de cifrado o empaquetado de software menos conocidos que utilizan los autores de malware. En función del conocimiento adquirido quizá puedan crearse nuevas reglas YARA o modelos predictivos que permitan detectar código ofuscado bajo nuevos métodos para, así, poder sospechar de su peligrosidad.

La simulación de ataque spear-phishing se realizó dentro de una red local controlada. Una posible línea de trabajo futuro consistiría en el uso de un dominio real y un servidor SMTP con salida a internet, de forma que fuese posible reproducir estos ataques hacia direcciones de correo controladas, pero funcionales. La finalidad de este trabajo tendría como objetivo experimentar diversas formas de alterar las cabeceras de los emails, evadiendo las medidas de seguridad que implementan clientes de correo populares como Gmail u Outlook.

El proceso de análisis dinámico de los binarios podría tratar de automatizarse a través de un sandbox como Cuckoo Sandbox o GFI SandBox. Asimismo, resultaría útil analizar otras familias de ransomware, tanto anteriores como posteriores, que guardasen similitud con Ryuk. Por ejemplo, Hermes, que ya se ha mencionado en este TFM.

La herramienta HectorCryptoHooking tiene posibilidades de mejora para aumentar en complejidad y funcionalidades. Un futuro proyecto de mejora podría consistir en la monitorización de otras funciones criptográficas, incluidas la nueva Cryptography API: Next Generation y envío de datos hacia un SIEM o servidor remoto. El autor es consciente de que la herramienta puede impedir el uso de aplicaciones benignas que utilicen estas funciones, por lo que sería provechoso investigar sobre métodos de análisis estadístico para detectar

escritura de datos aleatorios en los ficheros o la generación de claves aleatorias que supere un determinado umbral.

Los modelos predictivos generados con los algoritmos Random Forest y XGBoost, podrían integrarse en un SIEM para que, en el proceso de monitorización del sistema de ficheros de los dispositivos conectados, extrajese las características necesarias de un archivo PE y realice una clasificación que ayude a prevenir una posible amenaza.

Como última propuesta, la experimentación con algoritmos de aprendizaje automático diferentes a los que se han estudiado aquí quizá mejoraran los resultados obtenidos en la fiabilidad de los modelos predictivos. En este sentido, también convendría conseguir un set de datos con más variedad de información, incluyendo ransomware más actual.

Bibliografía

Nihad, A. Hassan. (2019). *Ransomware Revealed, A Beginner's Guide to Protecting and Recovering from Ransomware Attacks*. (1 ed., pp. 14-15). Packt Publishing.

NakedSecurity by Sophos. (diciembre de 2017). *5 ransomware as a service (RaaS) kits – SophosLabs investigates.*

Recuperado el 2 de noviembre del 2021 de: https://nakedsecurity.sophos.com/2017/12/13/5-ransomware-as-a-service-raas-kits-sophoslabs-investigates/

Webb, N. (enero de 2022). *Remove STOP/DJVU Ransomware Virus.*

Recuperado el 15 de febrero del 2022 de: https://geeksadvice.com/remove-djvu-ransomware-virus/

Kochovski, A. (enero de 2022). *Ransomware Statistics, Trends and Facts for 2022 and Beyond*.

Recuperado el 16 de febrero del 2022 de: https://www.cloudwards.net/ransomware-statistics/

Purplesec. (2021). *2021 Cyber Security Statistics. The Ultimate List Of Stats, Data & Trends.* https://purplesec.us/resources/cyber-security-statistics/

Yan, W. Zhang, Z. Ansari, N. Revealing Packed Malware. *IEEE Security & Privacy, 6*(5), 65 - 69. https://doi.org/10.1109/MSP.2008.126

Monnappa, K. (2018). *Learning Malware Analysis*. (1 ed.). Packt Publishing.

Jones, M. (marzo de 2020). *Top 13 popular packers used in malware*.

Recuperado el 13 de noviembre del 2021 de:
https://resources.infosecinstitute.com/topic/top-13-popular-packers-used-in-malware

Kumar,V y Beggs,R. (2019). *Setting up a phishing campaign with Gophish*. En *Mastering Kali Linux for Advanced Penetration Testing*. (3 ed., pp. 279-347). Packt Publishing.

Centro Criptológico Nacional. (marzo de 2021). *Informe Código Dañino CCN-CERT ID-03/21*

https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/5768-ccn-cert-id-03-21-ryuk-ransomware/file.html

Bhabesh, R. (septiembre de 2021). *Detecting Conti ransomware – The successor of infamous Ryuk.*

Recuperado el 2 de junio del 2022 de https://www.logpoint.com/en/blog/detecting-conti-ransomware-the-successor-of-infamous-ryuk/

Salah, A. Marhusin, M. Sulaiman, R. (septiembre de 2019). Instrumenting API Hooking for a Realtime Dynamic Analysis. *International Conference on Cybersecurity (ICoCSec), 2019,* 25-26. https://doi.org/10.1109/ICoCSec47621.2019.8971017

Ashraf, A. Aziz, A. Zhoora, U. Rajarajan, M. Khan, A (octubre de 2019). Ransomware Analysis using Feature Engineering and Deep Neural Networks. arXiv. https://doi.org/10.48550/arXiv.1910.00286

Security Ninja. (abril de 2015). *Spoof using right to left override (RTLO) technique.*

https://resources.infosecinstitute.com/topic/spoof-using-right-to-left-override-rtlo-technique-2/

Ammann, C. (mayo de 2012). *Hyperion: Implementation of a PE-Crypter*.

https://www.exploit-db.com/docs/english/18849-hyperion-implementation-of-a-pe-crypter.pdf

Kleymenov, A y Thabet, A. (2019). *Mastering Malware Analysis*. (1 ed.). Packt Publishing.

Mohanta, A y Saldanha, A. (2020). *Malware Analysis and Detection Engineering: A Comprehensive Approach to Detect and Analyze Modern Malware*. (1 ed). Packt Publishing.

Barker, D. (2021). *Malware Analysis Techniques*. (1 ed.). Packt Publishing.

Security Ninga. (mayo de 2015). *Windows functions in malware analysis – cheat sheet.*

https://resources.infosecinstitute.com/topic/windows-functions-in-malware-analysis-cheat-sheet-part-1

Documentación oficial de Microsoft. (noviembre de 2021). *Formato PE*.

Recuperado el 15 de diciembre de 2021 de: https://docs.microsoft.com/es-es/windows/win32/debug/pe-format

Hybrid Analysis. (septiembre de 2019). *MtXtS.exe Incident Response*.

Recuperado el 7 de diciembre del 2021 de: https://www.hybrid-analysis.com/sample/c7040cdf95e51827dbe6305e9c915dbd015a4de0fbd8f292c45b24b51ef37a5c

Documentación oficial de Microsoft (2022). *Rutas de acceso UNC*.

Recuperado el 3 de julio del 2022 de: https://docs.microsoft.com/es-es/dotnet/standard/io/file-path-formats

Documentación oficial de Microsoft. (mayo de 2022). *Schtasks.exe*.

Recuperado el 14 de julio del 2022 de: https://docs.microsoft.com/es-es/windows/win32/taskschd/schtasks

G-Data Security Lab. (junio de 2019). *Code-Signed malware: What's all the buzz about? Looking at the "Ryuk" ransomware as an example*.

https://www.gdatasoftware.com/blog/2019/08/35046-whats-all-the-buzz-about-looking-at-the-ryuk-ransomware-as-an-example

PassMark Software. (s.f). *CPU Test Suite Average Results for AMD Ryzen Threadripper 3990X*.

https://www.cpubenchmark.net/cpu.php?cpu=AMD+Ryzen+Threadripper+3990X

Hunter, B. (abril de 2020). *Stomping Shadow Copies - A Second Look Into Deletion Methods*. https://www.fortinet.com/blog/threat-research/stomping-shadow-copies-a-second-look-into-deletion-methods

Stanek, W. (2014). *Windows Server 2012 R2 Inside Out: Networking with TCP/IP - Understanding name resolution*. En *Windows Server 2012 R2 Inside Out Volume 2: Services, Security, & Infrastructure*. (1 ed., pp. 54-62) Pearson.

Admx. (s.f.). *Significado de claves de registro para Internet Settings\ZoneMap*.

https://admx.help/HKLM/Software/Policies/Microsoft/Windows/CurrentVersion/Internet%20Settings/ZoneMap

Mitre ATT&CK. (octubre de 2021). *Windows Management Instrumentation*.

Recuperado el 28 de junio del 2022 de: https://attack.mitre.org/techniques/T1047

Hexacom. (junio de 2015). *Basic intro to WMI and WQL queries*.

https://www.hexacorn.com/blog/2015/06/19/basic-intro-to-wmi-and-wql-queries

IANA - Internet Assigned Numbers Authority. (febrero de 2022). *IPv4 Multicast Address Space Registry*.

Recuperado el 5 de mayo del 2022 de : https://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml#multicast-addresses-1

INCIBE - Instituto Nacional de Ciberseguridad. (junio de 2021). *Ciberataques DrDoS basados en el protocolo mDNS*.

Recuperado el 27 de abril del 2022 de: https://www.incibe-cert.es/blog/ciberataques-drdos-basados-el-protocolo-mdns

Babkin, D. (octubre de 2020). *Malware researchers - Beware of GetProcAddress spoofing via manipulation of PE format in memory.*

https://dennisbabkin.com/blog/?t=malware-researchers-beware-of-getprocaddress-spoofing

MITRE|ATT&CK. (s.f.). *Deobfuscate/Decode Files or Information*.

Recuperado el 17 de marzo del 2022 de: https://attack.mitre.org/techniques/T1140/

Documentación oficial de Microsoft. (mayo de 2022). *Criptografía: Criptografía, CryptoAPI y CAPICOM*.

Recuperado el 30 de julio de 2022 de: https://docs.microsoft.com/es-es/windows/win32/seccrypto/cryptography-portal

Pinvoke. (s.f.). *A wiki for .NET developers*.

https://www.pinvoke.net/index.aspx

Cyble Inc. (julio de 2022). *"RedAlert", LILITH And 0mega Leading A Wave Of Ransomware Campaigns*.

https://blog.cyble.com/2022/07/12/new-ransomware-groups-on-the-rise

Bonaccorse, G. (2017). *Machine Learning Algorithms*. (1 ed.). Packt Publishing.

[1] Ullah, F. Javaid, Q. Salam, A. Ahmad, M. (2020). *Modified Decision Tree Technique for Ransomware Detection at Runtime through API Calls*. Hindawi. https://doi.org/10.1155/2020/8845833

[2] Lewis, E. Mlinarevic, T. Wilinson, A. (mayo de 2020). *Machine Learning for Static Malware Analysis*. nccgroup.

https://research.nccgroup.com/2021/06/07/research-paper-machine-learning-for-static-malware-analysis-with-university-college-london/

Aldid. (julio de 2017). *PE-Portable-executable*.

https://www.aldeid.com/wiki/PE-Portable-executable#Section_Characteristics

AnalyseUp. (s.f.). *Stratified Kfold Tutorial.*

Recuperado el 17 de septiembre del 2022 de: https://www.analyseup.com/python-machine-learning/stratified-kfold.html

ProjectPro. (julio de 2022). *How to drop out highly correlated features in Python.*

https://www.projectpro.io/recipes/drop-out-highly-correlated-features-in-python

Kana, M. (marzo de 2020). *How To Find Decision Tree Depth via Cross-Validation.*

https://towardsdatascience.com/how-to-find-decision-tree-depth-via-cross-validation-2bf143f0f3d6

Scikit-Learn documentación oficial. (s.f.). *Logistic regression (Version 0.24.2).*

Recuperado el 17 de mayo del 2022 de: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

Scikit-Learn documentación oficial. (s.f.). *Random Forest Classifier (Version 0.24.2).*

Recuperado el 17 de mayo del 2022 de: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Martínez, J. (septiembre de 2020). *Random Forest (Bosque Aleatorio): combinando árboles*.

https://www.iartificial.net/random-forest-bosque-aleatorio/

Morde, V. Anurag, V. (abril de 2019). *XGBoost Algorithm: Long May She Reign!*.

https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

Jain, A. (marzo de 2016). *Complete Guide to Parameter Tuning in XGBoost with codes in Python*.

https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python

Gavriel, H. (enero de 2019). *Cyberbit - New Ursnif Malware Variant, a Stunning Matryoshka*.

https://www.cyberbit.com/blog/endpoint-security/new-ursnif-malware-variant/

[3] JoeSandBox Cloud. (s.f). *Windows Analysis Report readerdc64_es_xa_install.exe.*

Recuperado el 21 de septiembre del 2022 de:
https://www.joesandbox.com/analysis/620495/0/html

Cantón, D. (enero de 2014). *RANSOMWARE III: Variante Lock Screen*.

https://www.incibe-cert.es/blog/ransomware-lock-screen

HP Wolf Security. (2021). *Threat Insights Report 1H - 2021*

HP Wolf Security Threat Insights Report H1 2021.indd